MEDIGS, UN SISTEMA SEGUR BASAT EN AGENTS PER AL DESCOBRIMENT I OBTENCIÓ DE DADES MÈDIQUES DISTRIBUÏDES. ESTUDI I DESENVOLUPAMENT.

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Abraham Martín Campillo i dirigit per Sergi Robles Martínez.

Bellaterra, Juny de 2007

El firmant, Sergi Robles Martínez, professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Abraham Martín Campillo

Bellaterra, Juny de 2007

Firmat: Sergi Robles Martínez

A tots aquells que signifiqueu molt per mi.

Agraïments

En primer lloc m'agradaria agrair l'esforç i recolzament continu del meu director de projecte, en Sergi Robles. Gràcies pels consells i ànims.

També voldria donar les gràcies a la resta de membres del grup SeNDA pel seu ajut. Sense oblidar als companys de projecte amb els quals he compartit moltes hores durant aquest últim semestre: Carlos, Franc, Jaume i Victor.

Gràcies a tots els grans amics que he fet durant aquests 5 anys de carrera i que m'emporto per sempre. En especial aquells que m'heu animat i ajudat sempre que ho he necessitat sense esperar res a canvi.

Donar les gràcies, en especial, a en Carlos Martínez, gran amic amb qui he compartit tants i tants moments durant aquest 5 anys. Gràcies per tot germà.

Agrair de tot cor a tots aquells que formeu part de la meva vida, tot el que faig és per i gràcies a vosaltres.

Índex

Índex de figures

Capítol 1

Introducció

El sistema sanitari és una de les primeres preocupacions de tot ciutadà, donat que del seu bon funcionament en depèn directament la nostra salut. En situacions d'emergència és imprescindible disposar de les màximes dades del pacient disponibles i de la coordinació més eficaç. Per això és molt positiu disposar d'una fàcil comunicació entre dispositius, centres i serveis. Aconseguir aquest objectiu amb dades en paper (com es tenien fins fa poc) és una tasca quasi impossible, però últimament hi ha un esforç important per informatitzar aquestos mateixos centres malgrat que encara hi manca un consens per compartir la informació entre centres. Aquesta informació digitalitzada habitualment només estarà disponible pel mateix centre mèdic que la genera, o bé, per un àmbit molt reduït de centres. Això, sumat a l'heterogeneïtat dels sistemes instal·lats als centres mèdics o hospitalaris i el seu enfoc centralitzat, dificulta les tasques de coordinació i compartició d'informació.

Tot país desenvolupat té una xarxa d'hospitals o centres mèdics públics i/o privats als quals hi pot anar una persona. Actualment quan una persona es troba a un hospital, el personal mèdic d'aquest centre de salut, generalment, només pot veure els anàlisis i proves que s'ha fet aquell pacient en aquest hospital. Encara que el pacient sigui habitual d'un altre centre mèdic i tingui totes les proves que s'ha fet emmagatzemades allà, el personal dels altres centres no tenen coneixement d'aquelles dades i proves. Això és degut a que no es solen compartir analítiques, proves, radiografies, etc, entre centres mèdics. No és gaire freqüent trobar una

xarxa mèdica en la que qualsevol personal mèdic autoritzat pugui accedir i veure tot l'historial del pacient i totes les seves proves quan repartides en varis centres de salut. Aquestes dades i proves, habitualment, només es guarden a la base de dades local del centre mèdic on s'han realitzat, i el seu accés només és intern. Això és el que cal millorar. ¿No seria molt positiu que un metge a l'hora de fer una diagnosi d'un pacient, o un tractament, pogués veure totes les proves i l'historial del pacient de tota la seva vida? Avui dia, no existeix una solució global per la recollida de dades mèdiques distribuïdes.

Actualment hi ha, i estan sorgint, moltes aplicacions utilitzant noves tecnologies, sobretot en el camp de la medecina. Una d'aquestes tecnologies són els agents. Els agents aporten moltes avantatges com el comportament intel·ligent utilitzant *behaviours*, preses de decisions, distribució de dades, migració segura de dades entre plataformes, entre d'altres; que van molt bé per les aplicacions mèdiques. És un paradigma relativament nou que no esta gaire estès però que comença a demostrar els seus avantatges respecte d'altres. Això ho demostra dos factors clau. D'una banda, totes les aplicacions que van sortint constantment basades en agents. D'altra banda, que darrere d'aquest paradigma es troba l'IEEE que s'encarrega d'estandarditzar tots els aspectes referents als agents. Un avantatge dels agents és, per exemple, la mobilitat. Utilitzant agents mòbils es poden crear esquemes d'informació distribuïda de manera ràpida, eficient i segura.

1.1 Objectius

Hem vist dues de les necessitats que té el món mèdic actual: la compartició d'informació i la cooperació més eficaç possible entre centres. Donat aquestes necessitats d'accés d'una manera segura a dades mèdiques distribuïdes, el nostre objectiu és satisfer aquestes necessitats i, per tant, contribuir a la millora del sistema sanitari. Per tal d'aconseguir-ho, després d'haver vist la tecnologia existent d'agents, confiarem en aquest sistema que està molt present al món mèdic i que és bo per crear esquemes d'informació distribuïda. Amb els agents volem desenvolupar un sistema capaç de resoldre el problema i portar a bon terme el projecte.

A més, ens basarem en un estudi previ sobre les mateixes necessitats [1].

Veiem el problema de prop amb un exemple. Si un pacient tingues una visita un dia determinat, a una hora concreta, actualment, el seu metge consultaria la base de dades on estan les dades del pacient 5-10 minuts abans de la visita o, fins i tot, durant aquesta. Miraria el seu historial i d'aquesta manera podria fer una diagnosi el més encertada possible o un tractament correcte. Però posem el cas de que aquell dia la base de dades no funcionés, que els sistemes estiguessin collapsats, o que la base de dades remota d'un altre hospital, on estan les seves dades, estigués en manteniment. El metge no podria consultar les dades del pacient.

Aquest problema el solucionaríem amb el nostre sistema. El nostre sistema s'encarregaria de tenir totes les dades del pacient sempre disponibles abans de la visita planejant la recollida de totes les dades abans d'aquesta. Així tindríem la visita a punt per quan estigui programada. En el cas d'haver-hi problemes de disponibilitat, el nostre sistema tindria temps per tornar a intentar-ho més endavant quan tots els centres mèdics estiguessin disponibles.

Però l'avantatge principal seria que tindríem una xarxa global de tots els centres mèdics on estarien disponibles per la resta de centres totes les dades de tots els pacients. D'aquesta manera el nostre sistema podria agafar totes les dades d'un pacient de tots els centres mèdics. Així l'historial del pacient a l'hora d'anar a qualsevol centre mèdic sempre estaria complert i disponible.

Un dels avantatges del nostre sistema seria que tindrem un sistema totalment automatitzat que faria totes aquestes tasques. A més, no suposaria cap canvi substancial als sistemes actuals implantats ja que s'integraria a dins d'aquests sense necessitat de canviar-los.

L'objectiu d'aquest projecte, per tant, és contribuir a la millora del sistema sanitari creant un historial clínic virtual. Tot i que hi ha previsió d'implantar el projecte a Portugal, l'objectiu és fer un sistema universal que es pugui implantar a qualsevol país amb una xarxa d'hospitals.

1.2 Què farem?

Per aconseguir el nostre objectiu, abans haurem de realitzar els següents passos:

1. Primer estudiarem i entendrem l'estudi previ fet a [1]. També estudiarem tot el rerefons actual del paradigma dels agents: com funcionen, qui hi ha darrere, com es programen, els estàndards que tenen i altres aspectes que ens facilitin la compressió d'aquesta tecnologia. D'aquesta manera entendrem millor tot el funcionament dels agents i podrem afrontar els següents passos coneixent molt bé la situació.

2. Seguidament farem un anàlisi de tots els requisits, funcionals i no funcionals, que se'ns planteja al projecte per saber quines seran les necessitats que tindrem.

3. Dissenyarem una solució tecnològica que sigui capaç de satisfer els requisits analitzats anteriorment utilitzant un sistema multi-agent.

4. Programarem una prova de concepte de la solució tecnològica dissenyada utilitzant el llenguatge i *framework* donat. D'aquesta manera portarem a la realitat una idea del nostre projecte, que serà funcional.

5. Validarem el disseny realitzat a traves de proves dutes a terme amb prototip programat. Les proves es realitzaran sota un ambient simulat, intentant que aquest sigui el més proper a la realitat (on està destinat que funcioni) possible.

6. Finalment, valorarem el sistema segons les proves fetes i determinarem les seves limitacions.

1.3 Estructura de la memòria

La resta de la memòria està estructurada de la següent forma:

Capítol 2 En aquest capítol veurem les definicions dels agents, els seus estàndards, especificacions, i la plataforma JADE. Aquest capítol ens servirà per poder entendre tota la resta de la memòria i situar-nos dins del context del projecte. També veurem les aplicacions actuals dels agents, sobretot aquelles dedicades a l'àmbit sanitari.

Capítol 3 En el capítol 3 farem un anàlisi detallat del projecte, identificant tots els requisits d'aquest, tant funcionals com no funcionals i poder veure així les necessitats que se'ns plantegen. També analitzarem la viabilitat tècnica, operativa, econòmica i legal del projecte.

Capítol 4 En aquest capítol dissenyarem una solució tecnològica que satisfaci tots els requisits descrits al capítol 3. Veurem els pros i contres de cada alternativa de cada aspecte necessari de dissenyar. Amb aquest disseny haurem de ser capaços de crear una aplicació que compleixi tots els requisits de l'anàlisi.

Capítol 5 En el capítol 5 veurem els detalls de la implementació de la prova de concepte de la solució dissenyada, els conceptes del disseny que hem escollit per tal de fer la prova de concepte, les proves que hem realitzat i una discussió sobre aquests resultats obtinguts: avantatges, inconvenients, problemes, millores, etc.

Capítol 6 Finalment, veurem les conclusions que podem extreure després d'haver fet el projecte, les millores a l'esquema i possibles línies d'ampliació futures.

Capítol 2

Agents i les seves aplicacions mèdiques

En aquest capítol descriurem els coneixements necessaris per entendre el projecte: el paradigma dels agents, de què es tracta, com s'utilitzen els agents, qui hi ha darrere, les utilitats que tenen, les aplicacions actuals, etc. Aquests coneixement seran necessaris per a que després de fer l'anàlisi de requisits puguem fer un bon disseny.

2.1 El paradigma dels agents

Un **agent** vist des del punt de vista del *software* no es defineix com una llista de mètodes i variables, sinó que el definim com uns comportaments determinats que seguirà. L'agent seguirà aquests comportaments amb un cert grau d'autonomia amb l'objectiu de complir els seus objectius. La definició exacta d'agent però, varia d'un autor a un altre. Tot i això hi ha certs punts on convergeixen la majoria d'autors i que per tant, podem dir, que definiria que és un agent. Aquesta convergència dels autors que podrem trobar a [9] és la següent:

Persistència El codi que s'executa en cada moment es decidit per el mateix agent en funció de l'objectiu que tingui, prenent la iniciativa i exhibint un comportament dirigit a complir l'objectiu.

Autonomia Els agents tenen capacitats pròpies de selecció de tasques, prioritats i presa de decisions sense intervenció humana.

Habilitat social Els agents són capaços d'interactuar amb d'altres agents mitjançant un tipus de llenguatge de comunicació entre ells, podent establir coordinació i col·laboració entre ells per executar una tasca.

Reactivitat Els agents perceben l'entorn, ambient i context en el qual es troben i reaccionen als seus canvis.

Quan un conjunt d'agents interactuen entre ells poden crear un **sistema multiagent**. Normalment tota la plataforma busca un objectiu global aconseguit pels objectius de cada agent.

Els **agents mòbils** són agents que es mouen a d'altres plataformes per continuar la seva execució al destí, portant amb ells tot el seu comportament i, si interessa, l'estat intern.

2.2 L'organització darrere els agents. Especificacions i estàndards

FIPA (*Foundation for Intelligent Physical Agents*) és un comitè de l'IEEE des de 2005 que promou la tecnologia basada en agents y la utilització dels seus estàndards per la interoperabilitat amb d'altres tecnologies. Seguir els estàndards de l'IEEE és una bona pràctica ja que ens proporcionaran compatibilitat amb d'altre programari o tecnologia que segueixi els mateixos estàndards.
FIPA té molts estàndards creats per al desenvolupament d'agents i els seus sistemes, tal i com es pot veure a [10]. Explicarem sense entrar en gaire detall de què tracten els principals estàndards definits per FIPA.

Una plataforma d'agents basada en l'estàndard FIPA inclou, entre d'altres agents, l'AMS i el DF. L'AMS (*Agent Management System*) és un agent que s'encarrega de registrar tots els agents que hi ha a la plataforma. Per fer un símil amb la realitat podríem dir que són unes pàgines blanques d'aquella plataforma,

on podem veure tots els agents que hi ha i la seva direcció (ex: agent@ccd-pr2:1009/JADE) amb la qual podrem contactar amb ells (com el telèfon de les pàgines blanques). També disposarem del DF (*Directory Facilitator*), tot i que serà opcional (podrà ser-hi o no). Aquest agent registra tots els serveis oferts pels agents d'una plataforma. A més, tal i com es fa amb els DNS, es pot federar i arribar a formar una xarxa de DFs on es pugui veure tots els serveis oferts de tots els agents de totes les plataformes. Per fer un símil com amb l'AMS, en aquest cas ens trobaríem davant d'unes pàgines grogues on s'ofereixen serveis i la direcció amb la qual hem de contactar per obtenir-lo.
Els agents es contacten i comuniquen entre ells mitjançant missatges ACL, també definits per FIPA, que estandarditza què han de contenir i com han de ser. Aquests missatges es transmetran mitjançant el MTS (*Message Transport Service*), un servei que proveu la plataforma per l'intercanvi de missatges entre els agents que es troben en aquesta i entre agents en plataformes diferents. En aquest segon cas, utilitzant també el MTP (*Message Transport Protocol*) i un sobre pel missatge ACL indicant la informació de transport (figura 2.1).

2.3 La plataforma JADE

De les moltes plataformes que existeixen, utilitzarem **JADE** (*Java Agent DEvelopment Framework*). JADE (figura 2.2) és un entorn de desenvolupament de codi obert basat en Java que implementa l'estàndard FIPA i ens permet crear sistemes amb agents segons aquestes especificacions. gràcies a que utilitza Java, serà independent de la plataforma *hardware*, nomes ens caldrà tenir la màquina virtual disponible pel sistema on el vulguem implantar. La plataforma conté un contenidor principal que és el que farà d'interfície amb l'exterior i pot contenir més contenidors no accessibles directament des de fora però funcionals, dintre de la mateixa plataforma. L'avantatge de tenir més d'un contenidor serà la de poder tenir una plataforma en varies màquines, situant un o varis contenidors a cadascuna d'elles. El contenidor principal serà el que contindrà l'AMS. La conjunció lògica de tots els contenidors serà la plataforma. Cada plataforma conté els agents

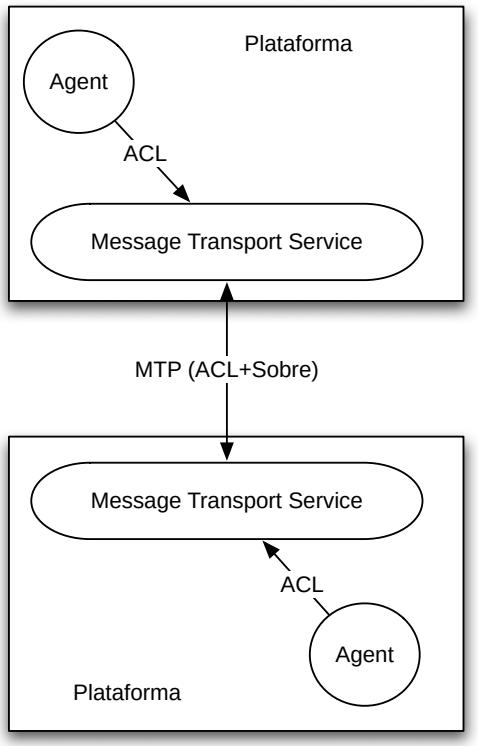


Figura 2.1: Message Transport

comentats en l'apartat anterior: AMS i, opcionalment, DF.

A JADE els agents utilitzen *Behaviours*, en català comportaments, que són classes que contenen accions que faran els agents sota unes determinades circumstancies. Un agent pot tenir més d'un *Behaviour* cadascun amb una tasca determinada. Aquests s'executaran a través d'una cua que en determinarà l'ordre d'execució depenent si aquests comportaments estan en estat *sleep* o no. Per modelar diferents comportament davant de diferents situacions no només podem programar el contingut dels *Behaviours* sinó que hi ha diferents tipus d'aquests que determinen la seva forma d'execució.

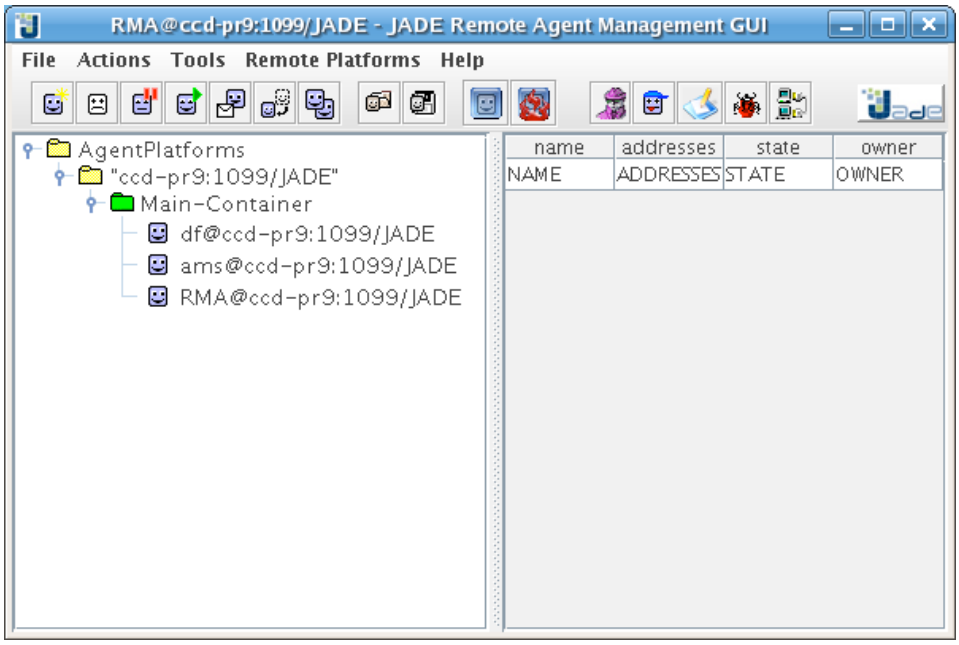


Figura 2.2: Imatge d'una plataforma JADE

2.4 Comunicació

Ara que ja hem vist com es comuniquen els agents, explicarem com es generen els missatges ACL i què es necessari per això.
Fent un paral·lelisme amb els humans, els agents poden tenir converses amb d'altres agents. Pel que nosaltres és una ona sonora que corre per l'aire, pels agents és un missatge ACL que es transmet per MTP. Tal y com nosaltres utilitzem un llenguatge y unes regles sintàctiques, semàntiques i gramaticals, els agents utilitzaran llenguatges i ontologies.

Els llenguatges són la gramàtica dels missatges i indicaran a l'agent com s'han de construir els missatges per enviar-los i com s'hauran d'entendre els missatges que arribin. És a dir, expliquen la codificació.

Els agents també utilitzaran ontologies (que són la semàntica del llenguatge) a l'hora d'intercanviar missatges, que ens serviran per entendre que és cada camp d'un missatge. Els dos agents implicats en una comunicació han de tenir les ma-

teixes ontologies per tal de que quan un rebi el missatge sigui capaç de passar-ho a camps: Un missatge és una cadena de caràcters i quan utilitzem una ontologia comuna, el que ens fa es passar aquesta cadena de caràcters a variables o camps coneguts per tots dos agents implicats en la comunicació. Per exemple, si enviem el missatge "La casa d'en Josep es blava" i utilitzem una ontologia de *característiques d'objectes*, el que farà l'agent quan rebi el missatge serà passar-ho a les variables *objecte*=casa, *propietari*=Josep i *color*=blava. Les ontologies, a més, indiquen subjectes i predicats d'una frase per tal de que tots dos agents implicats en la comunicació puguin fer una transposició directa del missatge (en cadena de caràcters) a contingut del missatge (variables). Vist des del punt de vista lògic podríem dir que si els dos agents no compartissin la mateixa ontologia quan un intercanvies un missatge amb un altre amb, per exemple el concepte "cadira", si el receptor no tingués la mateixa ontologia que l'emissor, no sabria de què li esta parlant.
Les ontologies es composen de predicats i/o accions. Una ontologia pot tenir més d'un predicat i/o accions però a l'hora d'enviar un missatge només podrà utilitzar un. Els predicats representen frases per informar sobre algun fet i les accions, com diu el seu nom, indiquen accions que volem que faci a qui li enviem (frases imperatives). Els predicats i les accions contindran *concepts*. Un *concept* indica un objecte i la seva descripció. Per tant, dins del *concept* tindrem molts camps que ens indicaran la descripció de l'objecte. Aquests camps podran ser de tipus primitius (String, Int...).

Totes dos en conjunció, els llenguatges i les ontologies, ens permetran que els agents entenguin els missatges que s'intercanviïn.

Els agents poden tenir més d'una conversa entre ells (amb un o més agents), per tant, haurem de diferenciar les converses d'alguna manera. Per això existeix el *conversation-id*, un identificador que ens permetrà separar diferents conversacions que estiguem mantenint amb el mateix agent.
Un altre camp rellevant al missatge és el camp "tipus", que ens indica el tipus de missatge. Aquests poden ser, per exemple, missatges de *request*, *propose*, *inform*, etc. Això ens informarà de quina és la intenció del missatge: informar, fer una

petició, una proposta, etc.

A més de tot això també existeixen protocols que ens indiquen en cada cas quin tipus de missatge hem d'enviar. Per exemple, si volem fer una petició, primer enviarem un missatge de petició, el receptor ens contestarà amb un missatge d'acceptació o negació i, si s'accepta, finalment el receptor ens enviarà un missatge amb el resultat d'aquesta petició. Existeixen diverses especificacions de protocols. Totes elles es poden consultar a la pàgina web de FIPA [10]. Seguint aquests protocols estàndards podrem interactuar amb qualsevol altre agent que segueixi l'estàndard i entengui la nostra ontologia i llenguatge.

2.5 Funcionament intern

Els agents poden disposar d'un o varis comportaments (*behaviours*) que s'executaran de forma seqüencial. Els comportaments estaran a una cua d'execució i aniran executant-se un darrere un altre. Quan un comportament acabi d'executar-se (un *return* al mètode *action*) es situarà al final de la cua. Els comportaments tenen estat. És possible que un comportament no s'executi si, per exemple, aquest està en estat de *sleep*. Quan un comportament acabi d'executar-se del tot, és a dir, el seu estat sigui *done*, aquest comportament sortirà de la cua i serà eliminat. També podem afegir nous comportaments durant l'execució de l'agent, els quals s'introduiran automàticament al final de la cua d'execució.

2.6 Migració

Que vol dir migrar? Doncs que un agent suspèn la seva execució per continuar-la després en un altre entorn d'execució. Això s'aconsegueix a baix nivell (es a dir, a nivell de codi) parant l'execució de l'agent en la plataforma origen. El seu codi font i estat es copia a la plataforma destí (serialitzant-lo, enviant-lo i deserialitzant-lo al destí) i una vegada copiat es reprèn l'execució de l'agent i es comprova que sigui correcte. Una vegada comprovat es destrueix la copia que ha quedat en la plataforma d'origen i ja tenim un agent migrat. Això és el que s'anomena migració

inter-plataforma. A més, hi ha una restricció: totes les classes que utilitzem als agents mòbils hauran de ser serialitzables per tal de que l'agent pugui migrar.

Existeix un altre tipus de migració, la migració intra-plataforma. Aquesta consisteix en que els agents migren dins d'una mateixa plataforma a diferents contenidors dintre d'aquesta. Aquesta migració existeix a JADE des de fa temps però la migració inter-plataforma només existeix des de 2006 a JADE gràcies a l'*add-on* JIPMS [11]. Aquest servei engegat amb la plataforma de JADE ens permet que els agents migrin entre plataformes que tenen aquest servei activat.

2.7 Seguretat

Els missatges entre plataformes que van per la xarxa es transmeten en clar, sense xifrar. Per aquest motiu és necessari xifrar els missatges que tinguin com a destí altres plataformes o contenidors (si aquests estan situats a d'altres màquines). Això és necessari per mantenir la privacitat, autenticitat, disponibilitat i integritat de les dades que s'intercanvien els agents. Si un tercer pugues canviar les dades, el sistema no ens seria de cap utilitat.

2.8 Aplicacions mèdiques

Actualment hi ha moltes aplicacions que fan ús dels agents a l'àmbit mèdic, només hem de veure [12], un número complet de la revista de referència en l'àrea, *IEEE: Intelligent Systems*, dedicat en exclusiva a aplicacions mèdiques utilitzant agents. Totes aquestes aplicacions utilitzen l'avantatge de l'aplicació de la intel·ligència artificial dels agents a dins de les aplicacions. Uns exemples d'aquestes aplicacions basades en agents són: aplicacions de tele medicina, decisions "intel·ligents", aplicacions per l'ajuda de pressa de decisions mèdiques, interconnexió i comunicació entre aparells mèdics, aplicacions d'urgències mèdiques (els agents preparen l'escenari per l'arribada d'un pacient), etc. Per tant, podem veure que les aplicacions utilitzant agents són una realitat i que hi ha aplicacions futures que estan per venir. Tot i això són molt poques les aplicacions que utilitzen la mobilitat

en aquesta tecnologia. En el nostre projecte ens centrarem en utilitzar i aprofitar aquesta mobilitat que ens ofereixen els agents, per tal de treure-li el màxim profit. A través dels pròxims capítols anirem desenvolupant mica en mica tot el projecte fins a arribar a tenir un disseny finalitzat així com un prototip que ens servirà per poder veure com es comporta el sistema dissenyat i poder treure conclusions.

Capítol 3

Anàlisi de Requisits

En aquest capítol veurem un anàlisi de requisits necessari pel nostre projecte realitzant seguint les especificacions IEEE indicades al document STD 830-1998 [4]. El nostre projecte pren com a base la proposta presentada a [1] i estén aquesta proposta a d'altres conceptes necessaris per la implementació del projecte. A través dels diferents apartats anirem veient els requisits necessaris (tant funcionals com no funcionals). Finalment, també veurem un estudi de la viabilitat del projecte.

3.1 Descripció Global

Després de comentar a la introducció quin és el context, la necessitat del projecte, etc, en aquest capítol veurem com volem que sigui el seu funcionament.
Voldrem un sistema que s'integri dins de l'actual sistema als centres mèdics. Aquest sistema haurà de recollir tots els nous esdeveniments clínics que es produeixin relacionats amb un pacient i, autònomament, anar a trobar totes les referències possibles d'aquell pacient (l'historial més complet possible) a tots els centres mèdics disponibles (que tinguin instal·lat i funcionant un sistema compatible). Una vegada recollides totes les dades, el personal mèdic autoritzat podrà veure tota la llista de referències d'aquell pacient i podrà accedir a elles, si estan localment emmagatzemades, o bé fer una petició de cessió d'una còpia d'aquella referència a la plataforma (centre mèdic) d'on s'ha obtingut. Per fer tot això

utilitzarem agents, un requisit que se'ns demanava al projecte, que ens permetran muntar sistemes autònoms interactius entre ells. Uns avantatges, els dels agents, que hem vist ja al capítol anterior.

3.1.1 Interfícies del sistema

El sistema haurà de complir els estàndards FIPA per a la inteoperabilitat entre sistemes compatibles: comunicació a través de missatge ACLs, oferir serveis a través del DF, utilitzar protocols estandarditzats a FIPA... El sistema haurà de funcionar sobre la plataforma JADE, un requisit del projecte. Per tant, el llenguatge de programació serà Java, i necessitarem una màquina virtual Java per executar-lo.

3.1.2 Interfícies d'usuari

L'usuari haurà de disposar d'una interfície en forma web o en forma de programa compatible amb totes les màquines clients per on s'hauran d'introduir esdeveniments clínics i peticions de descarrega de referències. Amb aquesta interfície s'haurà de poder introduir nous esdeveniments clínics, així com també gestionar els existents. També haurem de poder veure si una petició de descarrega o de cerca de referències ja ha sigut atès per l'agent corresponent. Quan la petició de cerca hagi estat acabada s'ha de poder veure totes les referències relacionades amb el pacient associat a aquell esdeveniment clínic i l'usuari hauria de poder seleccionar aquelles que vulgui que siguin baixades localment per tal de poder examinar-les (proves, anàlisis, etc).

3.1.3 Interfícies Software

Com que haurà de funcionar sobre la plataforma JADE, l'única dependència en quant a *software* necessari serà la d'un Sistema Operatiu que tingui disponible una màquina virtual de Java (versió mínima 1.4). Això inclou Linux, Solaris, Mac OS X, Windows, Simbian, etc. Com també volem que els agents puguin migrar d'una plataforma a una altre també requerirem de l'*add-on* de migració. També serà necessari un lloc on guardar totes les referències que recol·lectin els agents,

els esdeveniments que es produeixin, etc. Per tant, el nostre projecte requerirà obligatòriament:

- Maquina virtual Java (versió mínima 1.4) per poder utilitzar JADE.
- Framework JADE per utilitzar els agents. [13]
- L'*add-on* de migració JIPMS per migrar els agents d'una plataforma a una altra. [11]
- En el cas d'utilitzar base de dades per emmagatzemar esdeveniments clínics, referències, etc serà necessari un sistema gestor de base de dades. O bé, qualsevol altre tipus de repositori on guardar-les, en cas de no escollir una base de dades.

3.1.4 Interfícies Hardware

El programa haurà de poder ser executat en la varietat més gran possible de màquines per tal de poder reduir al màxim l'impacte econòmic d'implantar el sistema. Com que la màquina virtual de Java esta disponible per múltiples plataformes *hardware* (x86, PPC, x86_64, i fins i tot dispositius mòbils com ara PDAs amb processadors ARM) tindrem una alta compatibilitat. A més tenim l'avantatge de l'homogeneïtat, gràcies a Java. L'agent podrà anar a plataformes que estiguin en llocs diversos, des de dispositius mòbils fins a Workstations.

3.1.5 Interfícies de comunicacions

Un dels requisits també és el d'haver de tenir com a mínim una interfície de xarxa amb connexió a Internet o xarxa interna d'hospitals (intranet), per tal de que els agents puguin saltar d'una plataforma a una altra. Aquesta interfície podrà ser tant una connexió per cable (Ethernet) com una connexió sense fils (Wi-Fi, Bluetooth...). En el cas de connexions mòbils, necesitarem d'un *add-on* de descobriment de xarxes i serveis.

3.1.6 Memòria

La memòria primària necessària vindrà donada per les necessitats del sistema en cada moment i l'objectiu d'aquella plataforma en concret. Dependrà de l'entorn on s'executi (Sistema Operatiu, programes executant-se a l'hora) i també dependrà del nivell d'escalabilitat del sistema. Si tenim una xarxa molt amplia, rebrem moltes peticions constantment, el que farà que el nostre sistema necessiti de més recursos Hardware. Pel que fa a la memòria secundària, dependrà de la mida de Base de Dades, així com la quantitat i mida dels historials continguts a la Base de Dades i el sistema de fitxers.

3.1.7 Característiques dels usuaris

Els usuaris d'aquesta aplicació seran, habitualment, professionals de la medicina amb un coneixement tècnic a nivell d'usuari. Això ens ha de fer de dedicar-hi esforç a l'usabilitat de l'aplicació, per tal de que sigui fàcil i intuïtiva.

3.1.8 Aspectes legals

Les lleis de protecció de dades protegeixen les dades personals en general però molt especialment les dades més sensibles com ara les polítiques, religioses... i en el nostre cas, mèdiques. La llei de protecció de dades espanyola actual aprovada al 1999 (LOPDCP - Ley Orgánica de Protección de Datos de Carácter Personal [14]), classifica les dades mèdiques com les dades amb més alt nivell de protecció. Es una llei bastant restrictiva en quant a totes les mesures de protecció i registre que s'han de posar per poder accedir a una dada. Segueix la directiva europea aprovada per la mateixa causa, de tal manera que s'ha de registrar cada accés que es fa a la dada o cada modificació produida a aquesta. A més aquest registre ha de tenir indicat qui ha fet l'accés. Això ho haurem de tenir en compte quan dissenyem l'aplicació. Haurem de saber qui vol accedir a la dada cada vegada que fem la consulta a la base de dades. Per tant, l'agent haurà de portar aquesta informació quan vulgui fer consultes.

3.1.9 Seguretat

La seguretat és un requisit indispensable. Les dades que s'intercanviaran són dades de vital importància, en les que és molt important mantenir la integritat i confidencialitat. Les dades sobre infermetats del pacient han de ser mantingudes en el màxim secret possible. Les dades que ens informen de tots els anàlisis i estat del pacient han de tenir la màxima integritat per la seva importància a l'hora de diagnosticar i prendre decisions. A més de tot això, no haurà de ser possible manipular o accedir a aquestes dades per persones no autoritzades.

3.1.10 Requisits

Aquí farem un recull d'altres requisits del projecte així com un resum de les que hem esmentat fins ara.

- Seguir els estàndards FIPA.
- Desenvolupar l'aplicació (sistema multi-agent) utilitzant el framework JADE.
- Hem de poder fer migrar els agents, per tant necessitarem JIPMS.
- El sistema multi-agent haurà de ser tolerant a fallades, ja que no podem permetre que un agent es quedi sense poder continuar enmig d'una execució i no ens retorni les seves dades. Haurem d'acabar la seva execució o intentar restaurar el seu estat anterior.
- El sistema haurà de complir lleis internacionals de protecció de dades i lleis locals (LOPD en el cas Espanyol) en cas de que aquestes siguin més restrictives.
- La interfície d'usuari haurà de ser fàcil d'utilitzar i no haurà de suposar un nivell d'aprenentatge alt per part de l'usuari.
- Per la seguretat s'haurà d'implementar el sistema proposat a [1], que està basat en [2], que ens prové del necessari: privacitat, autenticitat, disponibilitat i integritat.

- Els sistemes de fitxers podran ser diversos en quant a que no influirà en quant a la implementació del programa.

- En cas d'utilitzar base de dades, hauran de ser amb suport SQL (Oracle, MySQL, etc) per tal de que no varii la implementació de l'agent que consulta la BD i afavorir la portabilitat.

3.2 Estudi de la Viabilitat

En aquest apartat farem un estudi de la viabilitat del projecte en tres camps: el tècnic i operatiu, l'econòmic i el legal. En ells veurem els pros i contres del sistema en cadascun d'aquests camps i la possible implantació real d'aquestos.

3.2.1 Viabilitat tècnica i operativa

Com hem vist a l'apartat anterior, les necessitats per tal de poder instal·lar el nostre sistema són ben poques i per tant serà possible instal·lar-lo en quasi tots els sistemes existents als hospitals. Val a dir que els hospitals són molt reticents a l'hora de compartir les dades amb d'altres hospitals, per un motiu, perquè no veuen l'avantatge per a ells. Però l'avantatge és clar: fent un símil amb un sistema P2P o de compartició, on es dóna, però també es rep a canvi quan es necessita. A més, com més gran es faci aquesta xarxa i s'expandeixi a més centres mèdics més avantatges tindrà, ja que més dades disponibles hi haurà. Un altre punt important a comentar són que proves tals com radiografies, ecografies, escàners, etc. són arxius que ocupen molt degut a que es fan a molt alta resolució, amb molta qualitat (ppp) i amb una baixa o nul·la compressió per tal de no confondre una mala codificació de l'imatge (punt canviat de color per exemple) amb alguna possible malaltia, infermetat o cos estrany. Això necessitarà d'un gran ample de banda per fer la transmissió i d'un temps determinat per transmetre l'imatge. També hi ha un punt clau diferenciable: no és el mateix que d'altres centres mèdics puguin accedir a les dades que el teu hospital te emmagatzemades, que aquestos mateixos centres puguin saber quines dades tens emmagatzemades al teu centre. La segona forma

seria un pas inicial per poder implementar aquest sistema, ja que no comportaria cap por degut a que només estarien donant informació de quines proves d'aquell pacient al seu hospital però no del contingut d'aquestes. Això donaria informació a la resta de centres mèdics que podrien donar quelcom a canvi d'accedir al contingut de la informació que els interessi, per exemple, un sistema de crèdits o punts.

3.2.2 Viabilitat econòmica

La viabilitat econòmica és millor que comparada amb d'altres alternatives que proposen el mateix repte però amb deficiències respecte a aquest projecte (que comentarem al capítol de conclusions). El cost de desenvolupament (possibles actualitzacions i millores), proves i desplegament seria l'únic realment important. També caldria un manteniment continu però aquest ja es té actualment a tots els centres mèdics per als sistemes actuals, per tant, no suposa cap despesa nova. No caldria necessàriament destinar pressupost a comprar maquinaria nova, ja que es podria utilitzar l'actual, excepte casos especials on la maquinària sigues incompatible amb els requisits abans esmentats. Cal comentar també que gràcies a que disposarem de les proves mèdiques d'altres hospitals no caldrà realitzar proves repetides que són cares, ocupen la màquina o laboratoris durant el temps que duri i tenen un cost en personal que intervé. Tot això suposarà un estalvi considerable en, per exemple, la Seguretat Social (cas Espanyol).

3.2.3 Viabilitat legal

La viabilitat legal del projecte s'haurà de tenir en compte en cada país, segons les seves lleis. Les lleis de protecció de dades, sovint, protegeixen molt les dades mèdiques i indiquen de quina manera s'han d'accedir, guardar, etc, degut a que aquestes dades són de caire molt sensible. Quines dades es poden tocar i quines no, definir diferents nivells de seguretat, diferent nivell d'accessos, sistemes d'emergències (prioritzar la vida a la llei), sistemes d'auditoria, etc. L'objectiu serà adapatar-ho a les màximes competències posibles, és a dir, implementar les direc-

tives europees sobre protecció de dades, però com les lleis de cada país tindran aspectes diferents potser haurem de fer petites adaptacions per complir les lleis específiques de cada un.

3.3 Planificació temporal

Abans de començar l'anàlisi del projecte, i després d'haver fet l'estudi sobre els agents i la documentació aportada per tal d'endinsar-nos a dins del context del projecte, vam decidir fer una planificació temporal del projecte. A la planificació temporal vam calcular quan tardaríem en desenvolupar cada part del projecte. Vam decidir dividir-ho en cinc parts: Anàlisi, Disseny, Codificació, Integració + Test i Memoria + Benchmarking.

Vam pensar que la part que ens portaria més temps seria la de codificació, on trobaríem els problemes i alguns possibles errors de disseny. Per tant, va ser la tasca a la que més temps vam assignar: cinc setmanes. A la part d'anàlisi li vam assignar una setmana i a la de disseny dues. A la part de test i correcció d'errors, així com una possible integració amb d'altres projectes si es veia la possibilitat, li vam dedicar 3 setmanes. Per últim, l'escriptura de la memòria (documentació del projecte) així com preparar el codi del projecte per tal de poder fer un entregable, se li va assignar les dues últimes setmanes.

La divisió temporal de les tasques, així com el temps que està previst que durin cadascuna d'elles i el seu ordre, està reflexat en el diagrama de Gantt (figura 3.1) fet amb el programa *Gantt Project* [15].

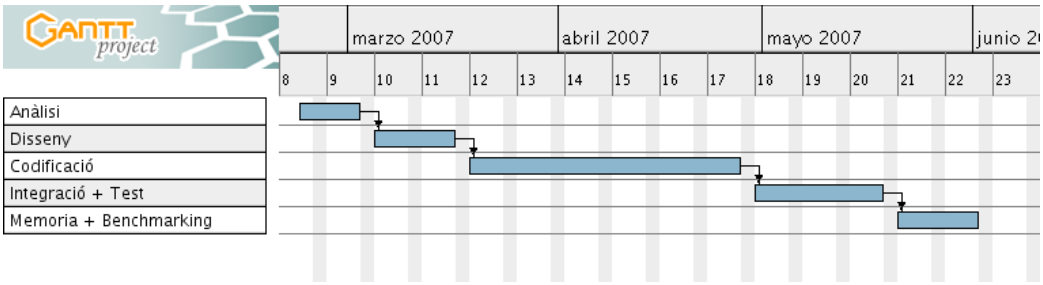


Figura 3.1: Planificació temporal

Capítol 4

Disseny de MedIGS

En aquest capítol començarem definint com serà l'esquema general de l'aplicació. En el capítol anterior hem vist el seu funcionament a mode d'anàlisi. En aquest veurem el seu funcionament a nivell de disseny. Dissenyarem quins agents hauran de fer quines tasques esmentades a l'esquema d'anàlisi i com es divideix l'esquema a nivell lògic. Discutirem sobre els avantatges i inconvenients de les alternatives que tenim per fer el disseny de cada punt de l'anàlisi (requisits) i triarem una d'elles. També veurem tots els aspectes tècnics derivats de la utilització dels agent: com utilitzar-los en el nostre cas concret i quina configuració hem de posar. L'objectiu d'aquest capítol és obtenir un disseny complet de l'aplicació a partir del qual es pugui extreure una implementació viable. Això últim serà exactament el que farem en el pròxim capítol.

4.1 Disseny global

En aquest apartat veurem com repartim les tasques definides a l'anàlisi. Cada plataforma haurà de tenir dues parts que es poden diferenciar lògicament: una part externa per la comunicació amb els agents que arribin des d'altres plataformes i una part interna per tal de gestionar els esdeveniments que es produeixin i l'historial del pacient, així com llençar i rebre (de tornada cap al punt inicial) agents.

La part interna gestiona els esdeveniments i disposarà dels següents agents:

Agent Gestor de Base de Dades (AGBD), Agent Intermediari d'Esdeveniments (AIE), Agent Programador d'Esdeveniments (APE) i Agent Local Intermediari de Documents (ALID).

AGBD L'Agent Gestor de Base de Dades consultarà periòdicament a la Base de Dades si existeixen nous esdeveniments. Una vegada obtinguts els nous esdeveniments, enviarà a l'AIE els esdeveniments nous de tipus cerca (*retrieval*) i a l'ALID els esdeveniments nous de tipus descarregar documents (*download*). L'AGBD marcarà com realitzats, per tal de no tornar-los a fer, els esdeveniments de cerca a la Base de Dades quan rebi la confirmació de l'AIE de que ha rebut s'ha fet la petició satisfactòriament. L'agent també marcarà com a realitzats els esdeveniment de descarrega quan l'ALID li enviï la confirmació de que s'han acabat de baixar. L'AGBD serà l'encarregat de rebre la llista de referències que hagin recollit els agents mòbils des de l'AIE i emmagatzemar-la a la base de dades, per tal de que es pugui accedir des de la interfície.

AIE L'Agent Intermediari d'Esdeveniments s'encarregarà de rebre les peticions de nous esdeveniments de cerca de l'AGBD. Aquests esdeveniments seran enviats a l'APE per tal de que llenci els agents mòbils oportuns. L'AIE s'esperarà a que els agents mòbils finalitzin el seu recorregut satisfactòriament i li enviïn els resultats. Una vegada tingui tots els resultats i l'esdeveniment de cerca hagi acabat satisfactòriament, l'AIE enviarà un missatge de confirmació a l'AGBD de que pot marcar l'esdeveniment a la base de dades com acabat. Juntament amb aquest missatge, també enviarà a l'AGBD la llista de referències que hagin recollit tots els agents mòbils relacionats amb aquell esdeveniment de cerca per tal de que es puguin emmagatzemar a la base de dades.

APE L'Agent Programador d'Esdeveniments rep de l'AIE els esdeveniments a processar. També rebrà la llista de plataformes o serveis disponibles d'un altre agent o d'un directori. Una vegada que tingui aquesta llista de plataformes i esdeveniments, l'APE llençarà un o varis agents mòbils per cada

esdeveniment, separant la llista de plataformes en cas de que enviem més d'un agent mòbil o amb tota la llista de plataformes en el cas d'utilitzar un de sol. Cada esdeveniment té associat a ell l'Identificador del pacient. Cada pacient té un Identificador únic a tota la xarxa d'hospitals.

MA Els Agents Mòbils seran manegats pel APE. Aquest els crearà, destruirà, configurarà i llençarà segons les seves necessitats en cada moment. Quan un agent mòbil visita una plataforma externa, es comunicarà amb l'Agent Remot Intermediari de Documents (ARID), al qui li demanarà totes les referències d'aquell esdeveniment. Quan els agents mòbils acabin el recorregut tornaran a la plataforma inicial (la plataforma de llançament) i enviaran la llista de referències que hagin recollit durant el recorregut a l'Agent Intermediari d'Esdeveniments (AIE).

Els agents mòbils es poden crear i destruir. El nostre esquema proposat serà el seguent: Tindrem un número d'ells pre-creats a la plataforma, és a dir, llestos per utilitzar-los. Aquest número podrà variar segons un paràmetre de configuració segons l'escalabilitat requerida. Per tant, aquests agents sempre estaran a la plataforma en mode *sleep*, o bé, treballant recorrent plataformes. També tindrem de dinàmics, és a dir, que es crearan nomes quan és necessitin: quan tinguem esdeveniments pendents però no ens quedi cap agent mòbil a la plataforma de llançament (tots ells estiguin ocupats recorrent plataformes). Els agents mòbils dinàmics els podrem destruir quan hagi passat un temps determinat sense que aquests hagin sigut utilitzats, o bé quan tornin de fer la tasca programada. Totes les dades de l'agent mòbil s'han de guardar en objectes serialitzables per tal de poder-se moure d'una plataforma a una altre.

ALID L'Agent Local Intermediari de Documents, com ja hem dit, rebrà referències de l'AGBD, i s'encarregarà d'accedir al repositori de documents remots i d'obtenir aquells documents que l'usuari hagi demanat de tota la llista de referències de documents associats a un pacient. Una vegada hagi acabat informarà a l'AGBD, el qual marcarà a la base de dades aquella referència

com a descarregada (*downloaded*).

La part externa disposarà d'un únic agent: Agent Remot Intermediari de Documents (ARID).

ARID L'Agent Remot Intermediari de Documents rebrà les peticions de referències a un Identificador de pacient des dels agents mòbils i aquest els hi donarà aquestes referències (direccions on l'ALID de la plataforma de llançament de l'agent mòbil haurà d'anar a buscar el document). L'ARID anirà a buscar aquestes referències a tots els AGBD de la seva plataforma que estiguin oferint el servei d'oferir referències. Això es degut a que podrà haver-hi més d'un AGBD en plataformes grans, com per exemple, parcs hospitalaris on hi hagin més d'una base de dades. Ja que, habitualment, en centres mèdics grans cada departament tindrà la seva pròpia base de dades per tal d'emmagatzemar les dades d'aquell departament.

Per veure gràficament aquesta explicació disposem de la figura 4.1, que ens ajudarà a entendre-ho millor. Per veure quines accions realitza cada agent disposarem del diagrama de casos d'ús de la figura 4.3. Tots els agents tindran similars diagrames de classes, aquest és el motiu de que només hàgim inclòs un (figura 4.2), com a exemple aclaridor d'aquest agent i de tots els altres. La resta d'agents tindran el mateix diagrama de classes, únicament canviat el nom de la classe (APE) i el nom dels seus comportaments (APEBe1, APEBe2, APEBe3). D'aquesta manera no volem fatigar al lector amb tot de diagrames iguals. Els dos diagrames de seqüència (figura 4.4 i figura 4.5) representen una seqüència completa dels dos tipus d'esdeveniments que hi poden haver. Com els diagrames UML no són adecuats per fer una bona representació dels agents, com migren (al diagrama de seqüència), la seva comunicació amb missatges ACL, els serveis que ofereixen, etc, hem decidit incloure altres esquemes que facilitaran molt la compressió completa del sistema en totes les seves vessants. Aquests esquemes els podrem anar veient durant el transcurs d'aquest capítol.

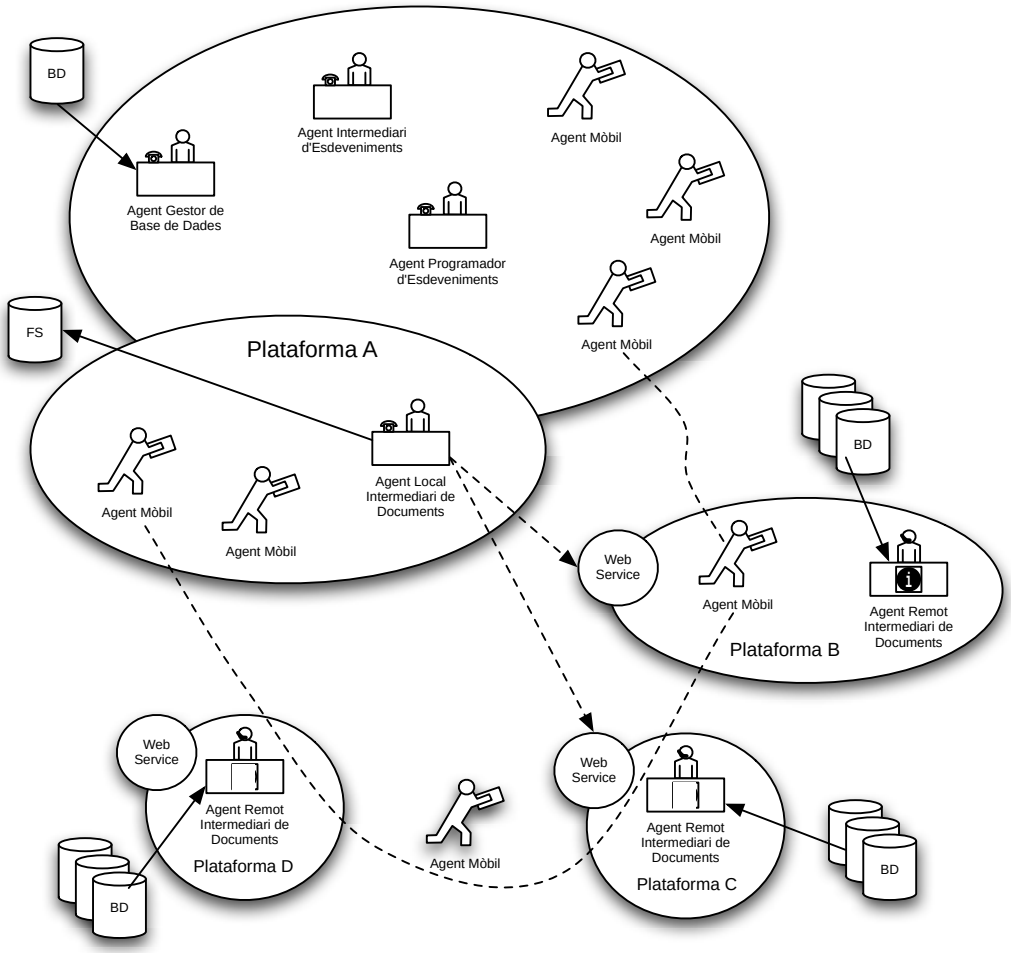


Figura 4.1: Esquema global

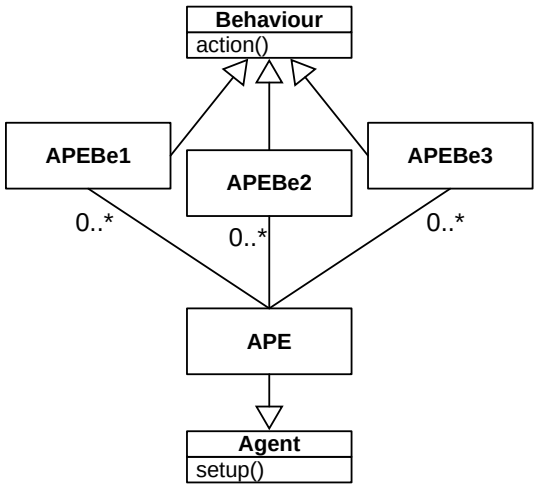


Figura 4.2: Diagrama de classes de l'APE

4.2 Interfície d'usuari (UI)

La interfície és un punt important dins d'un projecte, ja que és amb la que l'usuari es pot comunicar, interaccionar, veure resultats, etc i de la qual depèn molt la usabilitat. Ens centrarem, però, en com encabir la interfície dins d'aquest sistema multi agent autònom. Les dues principals opcions serien les següents:

1. Integrar la interfície a dins del sistema multi agent, dissenyant-la i implementar-la com un altre agent.

2. Separar la interfície del sistema multi agent i integrar-la dins del programa o interfície general del centre mèdic.

La interfície haurà de ser una aplicació web o un programa que pugui interaccionar amb la base de dades. A través d'ella l'usuari podrà inserir nous esdeveniments clínics: programar operacions, visites, proves, etc, que comportaran una cerca de les referències del pacient si es veu necessari. També podrà veure i fer una petició per descarregar parts de l'historial del pacient que es trobin a d'altres centres. Aquesta informació s'haurà obtingut a través d'un agent mòbil llençat per recollir a quin centre està situat cada part de l'historial del pacient relacionat amb algun

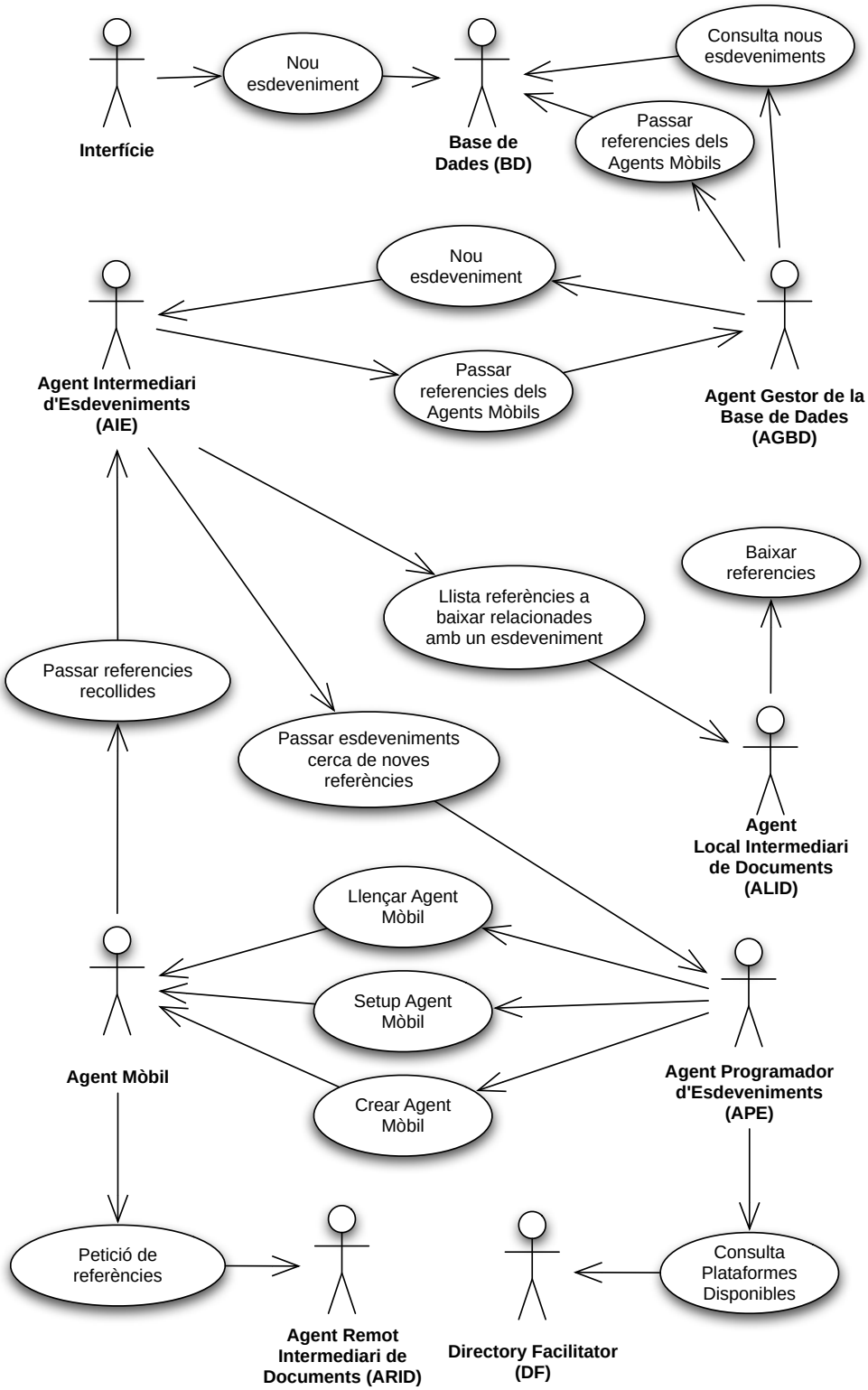


Figura 4.3: Diagrama de casos d'ús

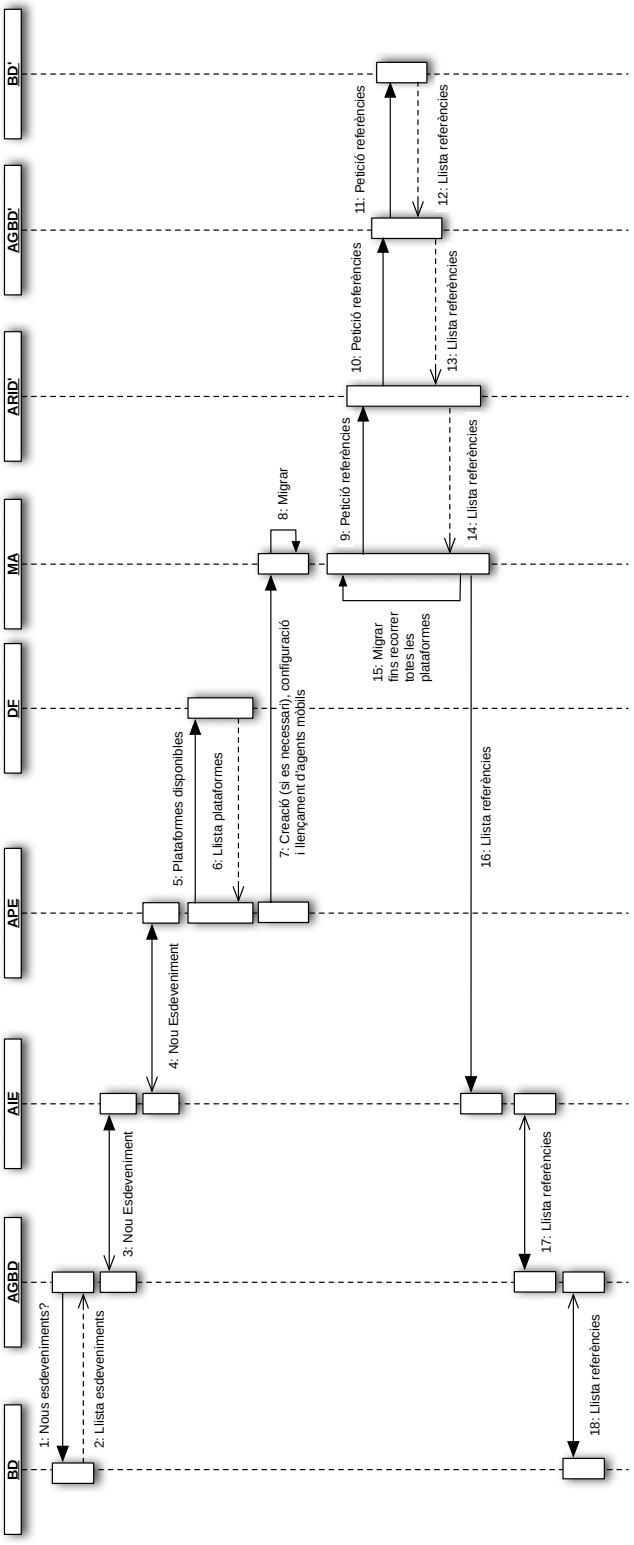


Figura 4.4: Diagrama de seqüència d'un esdeveniment de cerca de referències

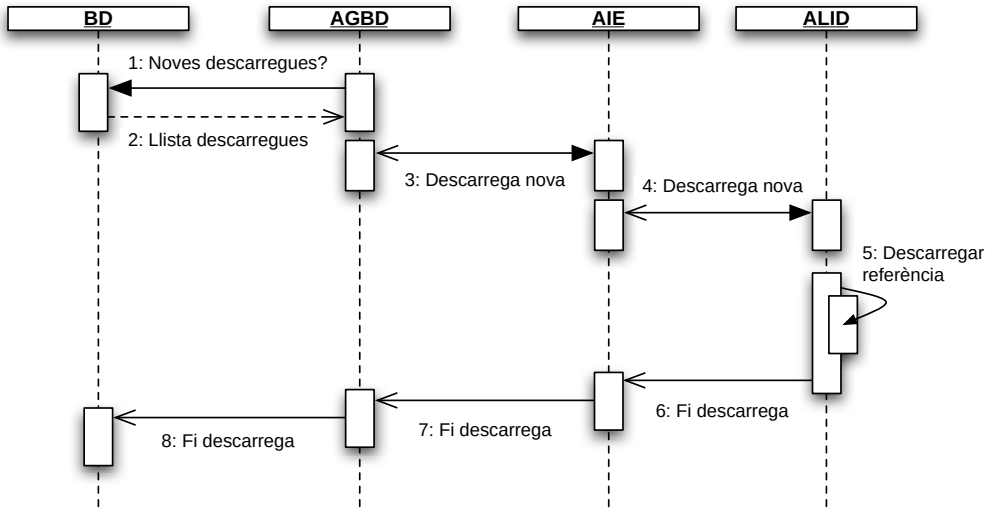


Figura 4.5: Diagrama de seqüència d'un esdeveniment de descarrega de documents

esdeveniment clínic. Aquestes dades i peticions quedaran inserides a dins de la base de dades per, posteriorment, ser processades.

Tenint en compte les dues opcions esmentades, veiem quines serien els avantatges i inconvenients de cadascuna d'elles:

Programa Dissenyar la interfície com un agent més dins del sistema multi agent ens permetria que el sistema estigues més compacte. Des d'aquesta opció veuríem la interfície com una part del sistema necessària perquè el sistema funciones. Però ben bé no es així: el sistema pot funcionar sense aquest component. Dit d'una altra manera, l'usuari no interactua a través de la interfície amb el sistema multi agent sinó que ho fa amb la base de dades. Quan inserim un nou esdeveniment ho fem a la base de dades, no al sistema multi agent, el sistema no se n'adona en aquell moment de que hi ha un nou esdeveniment. Posteriorment el sistema multi agent amb l'AGBD serà l'encarregat de consultar periòdicament la base de dades per tractar possibles nous esdeveniments.

Aplicació Web Aquesta opció ens permet integrar la interfície del nostre sistema

a la interfície general de tot el servei mèdic. La interfície no estaria dins el sistema multi-agent, però la manera de poder accedir a ella podrà ser més homogènia a la resta del sistema mèdic i estar situada en una millor posició des del punt de vista lògic. Degut al seu disseny, la interfície només haurà d'accedir a la base de dades, un recurs també extern al sistema multi agent. Per tant, es podria solucionar amb un servidor web amb suport PHP, o d'altres alternatives ja implementades a la resta d'interfícies del servei mèdic.

Per tant, vistes les dues opcions, degut a la seva facilitat de programació, bon resultat, estandardització i usabilitat, escollim la segona opció, l'aplicació web, vistes els seus avantatges respecte a la primera.

4.3 Base de dades

En aquest apartat discutirem com dissenyar el sistema gestor de la base de dades i la mateixa base de dades que necessitem. La base de dades estarà definida fora del sistema multi-agent. Això es degut a que com la interfície, que hem definit a l'apartat anterior, haurà d'accedir a ella constantment i aquesta està fora del sistema, hem decidit posar-la també fora. Es poden veure gràficament els conceptes que queden fora o dins del sistema multo-agent a la figura 4.14. Doncs bé, una vegada decidit això, com podem accedir a la base de dades? Tenim dues maneres:

1. Totes les comunicacions de tots els agents amb la base de dades passaran a través d'un agent gestor de la base de dades.

2. Cada agent que necessiti accedir a la base de dades accedirà directament a través de l'API de Java JDBC.

Començarem explicant la primera opció. El seu objectiu serà dissenyar un nou agent que s'encarregui de rebre totes les insercions o peticions a la base de dades de tots els agents que formen el sistema multi-agent i que només ell s'encarregui d'interaccionar amb la base de dades. És a dir, el gestor de la base de dades seria

un agent, l'Agent Gestor de la Base de Dades (AGBD).
A la segona opció, cada agent que necessités fer una inserció o petició ho faria directament utilitzant com a gestor de la base de dades les classe de l'API de Java JDBC.
L'avantatge de la segona opció seria la velocitat, degut a que no seria necessari passar a un altre agent (al AGBD) el que volem fer a través d'un missatge ACL. En canvi, des del punt de vista lògic dels sistema, estaria millor plantejada la primera opció, degut a que la base de dades és un recurs extern i dins d'aquest sistema multi agent seria un millor disseny que únicament un agent disposés d'ell. Així, tots els accessos a aquest recurs estarien canalitzats a través d'aquest agent amb comunicació amb tota la resta d'agents que composen el sistema. D'aquesta manera també milloraríem la independència. Ens seria molt més fàcil canviar el disseny de la base de dades, o la mateixa base de dades per un altre recurs, sense que tots els agents es veiessin afectats. Només hauríem de modificar l'AGBD.

Un altre aspecte que es va pensar va ser la forma en que notificàvem nous esdeveniments a la base de dades. Primer es va pensar d'una manera síncrona, és a dir, que l'Agent Intermediari d'Esdeveniments (AIE) fos el que s'encarregues d'anar preguntant al gestor de base de dades (AGBD) cada cert temps si hi havia un nou esdeveniment a la base de dades que necessités ser atès. Finalment, però, al haver escollit fer el gestor de la base de dades com un agent, el que s'encarregarà d'això serà aquest. Per tant, informar d'un nou esdeveniment a l'AIE per part del gestor es farà de forma asíncrona, sent l'AGBD l'encarregat d'anar preguntant periòdicament a la base de dades per nous esdeveniments i d'enviar aquests cap a l'AIE amb un missatge ACL quan es produeixin.

Per fer el disseny de la base de dades vam comptar amb l'ajuda de Pedro Marques, de la Universitat de Porto i co-autor de [1], que coneix com estan estructurades les base de dades als hospitals de Portugal. Vam rebre per part seva un arxiu Java on estaven els mètodes que feien insercions i consultes a la base de dades. D'allà en vam poder extreure com estava dissenyada la base de dades i amb la seva ajuda vam poder extreure les relacions i claus primàries i externes.
El disseny utilitzat és prou complexe i té en compte molts aspectes. La majoria

d'ells s'aparten de l'objectiu del projecte, degut a que és una base de dades professional especialitzada. Nosaltres la vam simplificar i només vam deixar aquelles taules i camps que utilitzaríem al nostre projecte, que a més, ens serviria per poder fer les nostres proves de disseny i implementació.

Abans d'explicar com s'ha dissenyat la base de dades, explicarem com es distribueix lògicament els dades dels pacient, les seves consultes, operacions, resultats, etc. La base de tot és el pacient. Un pacient té les seves dades personals on indica on viu, com es diu, entre d'altres i, a més, una clau identificadora única, de la que parlarem més endavant, que ens servirà per identificar al pacient de manera única a tota la xarxa d'hospitals. Cada vegada que el pacient ingressa a un hospital i se li programa una visita, una operació o una consulta amb un especialista, per anomenar alguns exemples, es crearà el que anomenarem un esdeveniment clínic. Aquest esdeveniment cínic contindrà tots els informes i proves (reports) que es facin en aquell esdeveniment clínic. Per exemple, si un pacient ingressa a un hospital perquè s'ha trencat una cama, aquell esdeveniment clínic contindrà les radiografies que li facin. A més, un conjunt d'esdeveniments clínics poden estar associats entre ells. És a dir, quan aquest mateix pacient torni a que li treguin el guix i li facin una segona radiografia per comprovar que l'os s'ha soldat, aquest esdeveniment clínic serà un altre esdeveniment clínic (l'esdeveniment clínic d'aquesta segona visita serà diferent de l'esdeveniment clínic de l'ingrés hospitalari) però tots dos estaran relacionats entre ells creant el que en direm un cas. Per tant, els casos seran un conjunt d'esdeveniments clínics sobre una mateixa infermetat o operació, etc. Per exemple, si ens han d'operar d'amigdalitis, totes les visites a l'especialista (un esdeveniment clínic per cadascuna), visita a l'anestesista, l'operació, visites posteriors per comprovar la recuperació del pacient, i demés, seran esdeveniments clínics dins d'un únic cas. La relació dels esdeveniments clínics en casos però serà una dada que no es tindrà en compte el nostre projecte ja que no es necessària. Així doncs, la base de dades prototip no tindrà aquesta relació. Per fer-ho una mica més entenedor, la figura 4.6 mostra de manera visual quina seria la jerarquia de pacients, esdeveniments clínics i reports.

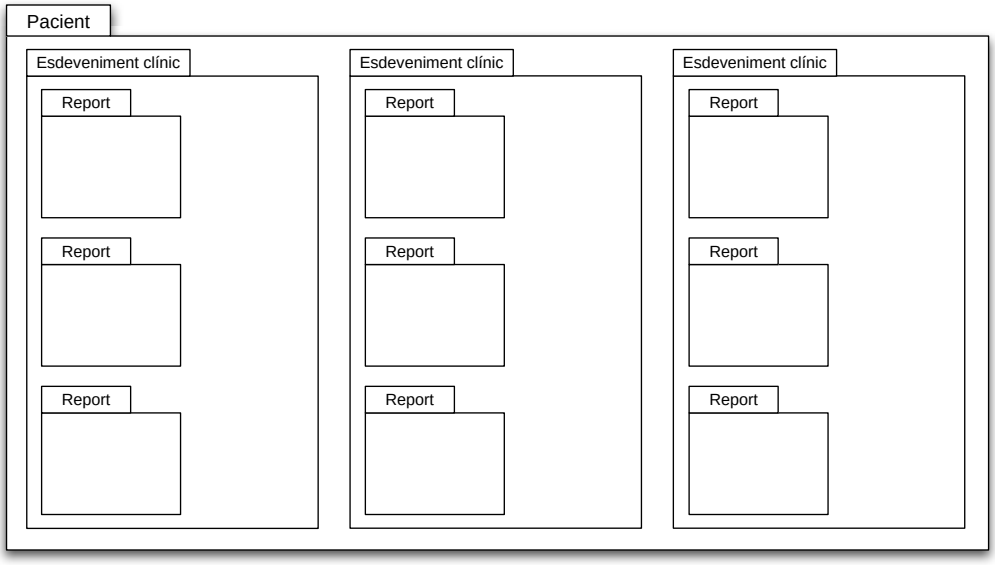


Figura 4.6: Esquema de pacients, esdeveniments clínics i reports

Com ja hem dit abans, l'esquema és més complexe però l'hem intentat simplificar. L'esquema complet no era l'objectiu d'aquest projecte, degut a que les bases de dades ja estaran implementades a cada centre mèdic. Cadascun tindrà la seva pròpia que no s'haurà d'adaptar, un dels grans avantatges d'aquest projecte, sinó que l'únic que s'haurà d'adaptar serà l'AGBD per a que agafi els camps necessaris de la base de dades corresponent. Aquests camps seran els que definirem més endavant a l'apartat d'Ontologies. Això ens aportarà uns avantatges clars: Cap canvi als sistemes actuals dels centres (podran continuar treballant igual sense haver de modificar la seva base de dades), i un sistema homogeni vist externament però amb heterogeneïtat interna.

La base de dades prototip que hem dissenyat i implementat per tal de fer proves, consta de quatre taules:

Pacient La taula pacient conté l'identificador únic de pacient del qual parlarem al següent apartat (NUM_SAN: Número Sanitari), així com les dades personals del pacient: el nom complet, sexe, adreça, data de naixement. Per mantenir la compatibilitat amb sistemes anteriors, és a dir, bases de dades

ja existents, hem tingut en compte que l'identificador únic (NUM_SAN) no fos la clau primària de la taula. La clau primària serà un altre camp (ID_PATIENT) ja que actualment cada centre mèdic tindrà la seva forma d'identificar els pacients.

Esdeveniment_Clínic Aquesta taula conté tots els esdeveniment clínics generats. Els esdeveniments clínics poden ser de diferent tipus com hem comentat abans. El camp de tipus numèric TIPUS_ESDEVENIMENT ens indicarà de quin tipus és: visita, operació, etc. A l'igual que la policia té els seus codis interns per indicar successos, a l'àmbit mèdic també. Per això aquest camp es de tipus numèric enlloc de cadena de caràcters. Si fos necessari hi podria haver una altra taula relacionant aquests codis amb unes cadenes de caràcters què expliquessin que significa cadascú. Aquest mètode d'identificar tipus d'esdeveniments cínics estalviarà possibles confusions i serà més senzill a l'hora de comunicar-ho entre serveis mèdics. A cada esdeveniment clínic tindrem el camp ID_PACIENT que ens relacionarà l'esdeveniment clínic amb un pacient (clau externa) i un camp ID_DOCTOR que ens indicarà qui és el personal mèdic encarregat d'aquell esdeveniment clínic. Aquest camp podria ser múltiple, és a dir, que hi hagués més d'un personal al càrrec, i a més, en el cas que existís una altre taula amb el registre de tot el personal mèdic de l'hospital es podria convertir en una clau externa. També tindrem el camp DATA_ESDEVENIMENT que és la data quan s'introdueix l'esdeveniment clínic i el camp DATA_CERCA que ens indicara a partir de quina data podem fer la cerca de dades relacionades a aquell pacient. També tindrem el camp CERCAT que ens indicarà si hem d'anar a cercar dades d'aquell pacient (en cas de que tingui valor fals) o no (en el case de que tingui valor cert) quan la data sigui posterior a DATA_CERCA.

Report En aquesta taula tindrem tots els reports que s'hagin creat en aquell centre mèdic, amb un identificador únic ID_REPORT, un camp ID_ESDEVENIMENT que ens relacionarà el report amb un esdeveniment clínic existent (clau externa), un camp DATA_REPORT que ens indicarà la data en què

es va crear el report, un camp ID_DOCTOR del mateix significat que el de la taula Esdeveniment_Clínic, i un camp URI que contindrà la direcció des de la qual es pot accedir a aquest report externament (per quan un agent mòbil busqui dades de reports i aquests posteriorment es baixin des d'una altra plataforma). També tindrà un camp DESCRIPCIÓ que contindrà una descripció del report però que no ha de ser necessàriament una diagnosi, sinó una descripció per tal de que personal d'altres centres puguin saber més específicament de quin tipus de report es tracta. Justament per això també incloem el camp TIPUS, de tipus numèric, que identificarà el tipus de report amb un codi de la mateixa manera que ho fa anàlogament el camp TIPUS_ESDEVENIMENT de la taula Esdeveniment_Clínic.

ReportExt En aquesta taula tindrem totes les referències que hagin recollit tots els agents mòbils durant les seves cerques a d'altres plataformes, aquesta taula tindrà tots els camps que també té la taula Report excepte el de ID_ESDEVENIMENT, ja que no podrem relacionar un report d'una altra plataforma amb cap esdeveniment clínic del nostre centre. A més d'aquests camps també tindrà un camp identificador únic que comentarem més endavant, un camp NUM_SAN (Número Sanitari) que serà l'identificador únic del pacient a qui pertany aquell report, un camp HOSPITAL que ens informarà a quina plataforma es va generar aquell report, un camp DESCARREGAT per informar si disposem d'aquest report al nostre File System (és a dir, ja l'hem descarregat), un camp DESCARREGAR per poder fer una petició de decarregar aquell report a la nostra plataforma, i així disposar d'ell, i un camp QUI que servirà per indicar qui demana que es descarregui aquell report, ja que s'haurà d'informar d'això al centre d'on baixem aquell report. Això és necessari per tal de complir amb la llei de protecció de dades, tal com s'explicarà més endavant.

A la figura 4.7 es pot veure l'esquema de la base de dades. En negreta figura la clau primària de cada taula i amb les fletxes es mostren les claus externes.

Es va buscar un sistema de base de dades lleuger, gratuït, senzill i que s'integrés bé amb Java per tal d'implementar aquesta base de dades. Es va trobar

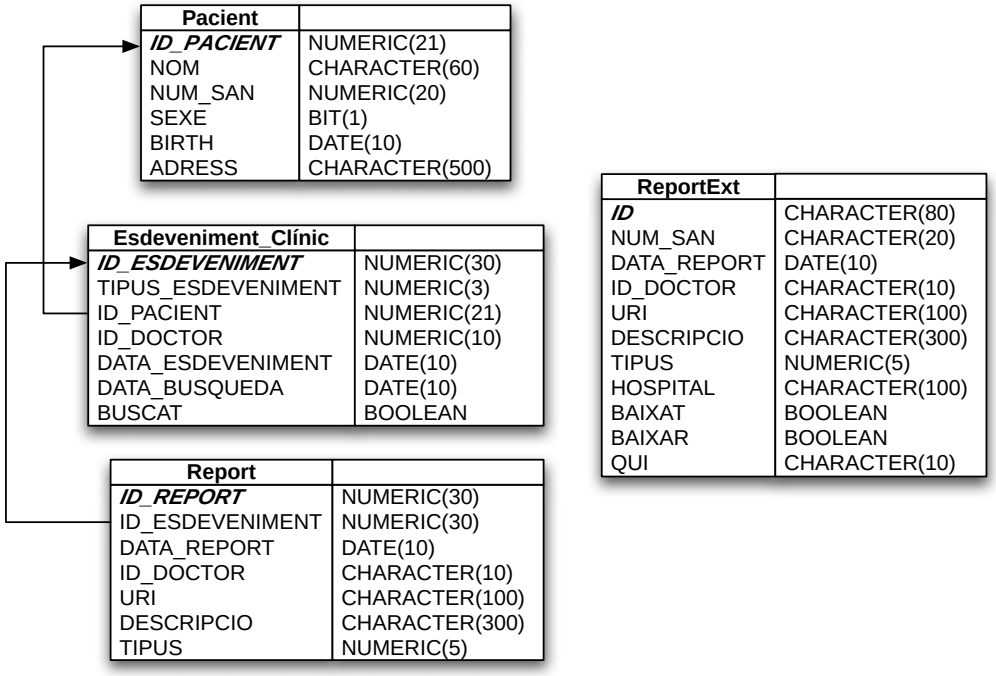


Figura 4.7: Esquema relacional de la base de dades

One$DB [5], una Base de Dades Java de codi lliure, que compleix els estàndards JDBC 3.0 i es SQL 99 compatible. A més, al ser Java és independent de la plataforma on l'executem.

4.4 Identificador únic de pacient (Número Sanitari)

Un problema que va sorgir quan vam estar dissenyant la base de dades va ser la manera d'identificar a un pacient únicament a tota la xarxa completa d'hospitals. Això comporta un problema greu ja que si totes les bases de dades no comparteixen el mateix número de pacient per cada pacient, seria impossible identificar a un pacient i poder buscar tots els seus reports a totes les plataformes. Si totes les plataformes estan situades a dins d'un mateix país el problema s'alleuja, ja que podrem utilitzar com a identificador únic el DNI del pacient, el seu numero de

la seguretat social, o qualsevol número identificador del qual es disposi a nivell de país. Però el problema real arriba quan l'escalabilitat supera fronteres, on el problema es fa difícil de solucionar sense que s'hagi de crear un estàndard per tal de definir com identifiquem cada pacient de manera única. Realment la escalabilitat no serà habitualment tant alta, degut a que no és corrent que un pacient canvii de país constantment per fer-se proves mèdiques i, per tant, a curt termini no es veurà com un problema. Tot i això, intentarem resoldre el problema a curt i mig termini. A curt termini serà fàcil resoldre, com ja hem dit abans, ja que les plataformes estaran situades dins d'uns mateix país. Però centrem-nos a mig termini en una situació propera com per exemple Europa, que està formada per diferents països però dins d'una comunitat. Quan diversos països adoptessin el sistema i volguessin compartir les seves dades necessitarien compartir el mètode en que identifiquen el pacient. Com fer-ho? Després d'investigar sobre el tema vam descobrir la targeta sanitària europea [6]. La targeta sanitària europea és una targeta que pot demanar qualsevol membre de la Unió Europea per identificar-se i poder rebre prestació sanitària a tota Europa. Conté un camp identificador únic que ens resoldria el problema de mig termini (figura 4.8).

Evidentment, el problema a nivell global és més complexe i es podria solucionar adoptant uns estàndards. Per exemple: utilitzar un identificador de país juntament amb un número identificador únic d'habitant d'aquell país. En el cas d'Espanya podria ser el DNI, per tant quedaria: IDESP42584213L. Tot i això seria necessari imposar un estàndard global, ja que la longitud del camp variaria segons el país i l'identificador de país hauria de ser únic. També s'ha de tenir en compte que hi ha països que no identifiquen de manera única als seus habitants.

Relacionat amb la identificació única també ens sorgeix el problema de com identifiquem un únic report a tot el món, per tal de que no hi hagi més d'un amb el mateix ID a dins de la base de dades. La solució aquí és més fàcil: ens bastarà amb concatenar l'identificador de plataforma (centre mèdic), que serà únic, juntament amb l'identificador del report que utilitza internament aquella plataforma. D'aquesta manera no hi haurà perill de que coincideixin IDs.

Figura 4.8: Targes sanitàries europees. Model europeu genèric (imatges superiors) i model espanyol (imatges inferiors).

4.5 Esdeveniments

Primer aclarirem dos conceptes que potser poden crear confusió. Durant la memòria parlem d'esdeveniments i d'esdeveniments clínics. Els esdeveniments clínics són aquells esdeveniments relacionats amb un centre mèdic, és a dir, les operacions, visites, entre d'altres, que es programen. Aquests esdeveniments clínics seran els que introduirem a través de la interfície d'usuari quan vulguem programar algun d'aquests esdeveniments clínics a un pacient. Els altres tipus d'esdeveniments, no clínics, seran aquells que es produeixin automàticament (obtenir referències) com a resultat d'haver introduït un esdeveniment clínic d'un pacient del qual no tenim dades actualitzades de referències del seu historial, o bé, aquells que es produeixin com a petició del personal mèdic autoritzat (descarregar referències). En tots dos casos, aquests tipus d'esdeveniments, no clínics, seran

esdeveniments des del punt de vista del sistema multi-agent, no des del punt de vista d'un centre mèdic. Una vegada aclarit aquest punt, veiem amb més detalls els tipus d'esdeveniments no clínics que hi poden haver:

1. Obtenir referències

2. Descarregar referències

Quan introduïm un ID de pacient a la interfície, aquesta ens mostrarà la llista de referències (documents que tenim o estan a d'altres hospitals) que tenim relacionades amb aquest ID a la nostra Base de Dades. Llavors, podrem crear dos esdeveniments possibles. El primer consistirà en obtenir noves referències d'aquest pacient, es a dir, enviar un agent mòbil a recórrer totes les plataformes per obtenir noves referències. El segon consistirà en marcar aquelles referències que no tenim en local, es a dir, estan emmagatzemades a d'altres hospitals, per tal de que l'Agent Local Intermediari de Documents (ALID), s'encarregui de descarregar-les. Per això hi hauran tres tipus de referències: Les primeres (marcades a la interfície d'usuari de color verd) seran les referències a documents que tindrem en local, les segones (marcades de color groc) seran les referències a documents emmagatzemats a d'altres hospitals però que podrem llegir (portar una copia a local per accedir-hi) i les terceres (marcades en vermell) seran les referències que hi ha a d'altres hospitals però que no podrem accedir a elles.
Val a dir, que el primer esdeveniment es podria generar automàticament si el sistema multi-agent ho cregués oportú. Aquest cas es produiria quan, per exemple, es programa un esdeveniment clínic a un pacient del qual no tenim cap referència del seu historial mèdic, o bé, aquesta llista de referències està desactualitzada: fa molt temps que es van anar a buscar i ara podrien haver-hi moltes més.

En el apartat de comunicació podrem veure com s'informen els agent entre ells de nous esdeveniments i quins són els camps de la base de dades que s'intercanvien.

4.6 Emmagatzemament

Totes les referències externes que descarreguem a través de l'ALID, les emmagatzemarem en un sistema de fitxers local de manera indexada, per tal de que poder accedir-hi sempre que ho necessitem i no haver-ho de tornar a descarregar. La indexació anirà a càrrec del sistema de fitxers, prèviament configurat per fer aquesta tasca. També, però, es podrà utilitzar una alternativa al mètode que hem escollit, el sistema de fitxers, que serà la següent: emmagatzemar els reports descarregats a dins de la base de dades. En aquest cas, necessitaríem una base de dades preparada per tenir emmagatzemat dades de gran mida.

4.7 Serveis

Per tal d'enviar missatges ACL d'un agent a un altre és necessari conèixer la seva adreça. Un exemple d'adreça seria: ams@ccd-pr2.uab.es:1099/JADE. Aquest mètode és estàtic per tal d'accedir a l'agent, és a dir, hem de conèixer el seu nom i la direcció del host (port inclòs) per tal de poder enviar-li el missatge. Es veu clarament que això no és viable a l'hora de programar, ja que els agents estaran a diferents plataformes de les quals no coneixem el nom. Una possible solució seria guardar en un fitxer de propietats el nom de les plataformes, però igualment seria un mètode ineficient, ja que s'hauria de modificar a mà i el nostre objectiu és obtenir un sistema completament automàtic. Diferenciarem dos casos, quan vulguem comunicar-nos amb un agent dins de la mateixa plataforma que nosaltres i quan vulguem migrar cap a una altra plataforma (hem de saber la seva direcció).

En el primer cas, el nom de la plataforma el podrem saber ja que ens ho dirà un mètode del nostre agent, però haurem de saber també el nom de l'agent. El nom de l'agent podria ser un requisit de l'aplicació, és a dir, els agents haurien de tenir el nom pel qual han estat dissenyats, però ho hem volgut fer més flexible i que els agents puguin tenir el nom que es vulgui. Però llavors quan vulguem comunicar-nos amb un agent, com per exemple amb l'APE, com sabrem el seu nom? Doncs utilitzant el *(Directory Facilitator)* explicat al capítol 2. Cada agent que ofereixi

un servei, és a dir, que hagi de rebre una comunicació per part d'un altre per iniciar un protocol, es registrarà al DF. Per tant, a l'hora de fer la implementació, tots els agents que compleixin aquestes condicions es registraran al DF indicant el nom, tipus, ontologia/es i protocol/s que fa servir el servei. D'aquesta manera ja no ens caldrà saber el nom dels agents. L'administrador del sistema podrà posar el nom que desitgi, ja que per trobar l'agent desitjat no utilitzarem el seu nom sinó una cerca del servei que busquem al DF. El DF ens retornarà el nom de l'agent i l'utilitzarem com a destinatari al missatge. Això encara ens dóna un avantatge més: si es vol canviar un agent per un altre amb la mateixa funcionalitat però que no pertany al nostre projecte, es podrà fer sempre que es registri al DF donant el mateix servei i utilitzant el mateix protocol estàndard FIPA.

A la figura 4.9 podem veure tots els serveis que ofereix cada agent a dins d'una plataforma. A cada servei de cada agent indicarem quins protocols i ontologies utilitza. L'explicació de cada ontologia i protocol la farem al següent apartat. A la figura 4.10 trobem quin són aquells agents que utilitzen quins serveis.

El segon cas, migrar d'una plataforma a una altra, és a dir, saber el nom de les demés plataformes, és més complicat i contempla varies alternatives. Necessitem un mètode per registrar totes les plataformes i que aquestes puguin accedir-hi per saber quines hi ha disponibles.
Una primera possible solució seria la d'un directori de plataformes. Es podria muntar de la següent manera: podríem tenir una plataforma general on es registressin totes les plataformes al seu DF. Cada plataforma s'hauria de donar d'alta en aquesta plataforma general per a totes i quan necessités saber la llista de totes les plataformes en funcionament miraria aquest DF general. Per fer això ens caldria un agent que només tingues aquesta funcionalitat, la de registrar-se a si mateix al DF general perquè així d'aquesta manera la resta de plataformes pugin saber la direcció de la plataforma en la que està.
Una altra manera seria la de instal·lar a cada plataforma un client LDAP [21] i que es registres a un directori general LDAP on quedaria una llista de totes les plataformes disponibles. En aquest cas tindrem l'avantatge de la seguretat, ja que el servidor LDAP també conté la llista de claus públiques de cada plataforma per

poder-les utilitzar a l'enviar un missatge, encara que de manera anàloga si utilitzem el primer mètode proposat podríem utilitzar algun camp del registre del servei al DF per emmagatzemar la clau pública de la plataforma.

4.8 Comunicació

Al capítol 2 ja vam veure com es comunicaven els agents i el significat de les ontologies i dels protocols. En aquest apartat descriurem quines són les ontologies i els protocols que utilitzarem. Un aspecte a tenir en compte és que serà necessari posar un temps màxim d'espera als missatges per a la tolerància a fallades i el control d'errors. D'aquesta manera també podrem fer reintents. En aquest apartat podrem veure l'estructura dels dos tipus d'esdeveniments disponibles: peticions de cerca i peticions de descarrega. Els podrem veure al subapartat d'ontologies: l'estructura de l'esdeveniment de cerca serà el *concept* DOCUMENT_RETRIEVAL_DESCRIPTION, contingut dins del primer predicat de l'ontologia d'esdeveniment de cerca. L'estructura de l'esdeveniment de descarrega serà el *concept* DOCUMENT_DOWNLOAD_DESCRIPTION, dins de l'acció de l'ontologia d'esdeveniment de descarrega.

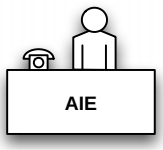
4.8.1 Protocols

Per mantenir una conversació, en el nostre cas utilitzant un protocol per tal de fer-lo estàndard, hem d'utilitzar una ID única per identificar la conversa. Tant si l'iniciem nosaltres, creant nosaltres aquest ID, com si la inicia l'altre part, haurem de respondre aquells missatges que hem rebut utilitzant el mateix ID i, a més a més, seguint el tipus de missatge que indica el protocol. Això és degut a que un agent pot mantenir més d'una conversa al mateix temps amb diferents agents o inclús amb el mateix agent. Per tant, és necessari l'identificador per diferenciar una conversa d'una altra. Aquest paràmetre s'anomena *conversation-id*.

En qualsevol moment d'un protocol, el receptor del missatge pot respondre a l'emissor d'aquest que no entén el missatge enviant un missatge *not-understood*.

Nom Servei	Ontology	Protocol
UnloadRefernces	document-retrieval-ontology	fipa-propose
SearchReferences	document-retrieval-ontology	fipa-request

Nom Servei	Ontology	Protocol
DocumentRetrieval	document-retrieval-ontology	fipa-propose



Nom Servei	Ontology	Protocol
DocumentUnload	document-retrieval-ontology	fipa-propose
NewEvent	document-retrieval-ontology document-download-ontology	fipa-propose

Nom Servei	Ontology	Protocol
DocumentRetrieval	document-retrieval-ontology	fipa-request

Nom Servei	Ontology	Protocol
DocumentDownload	document-download-ontology	fipa-propose

Figura 4.9: Llistat de serveis per agent

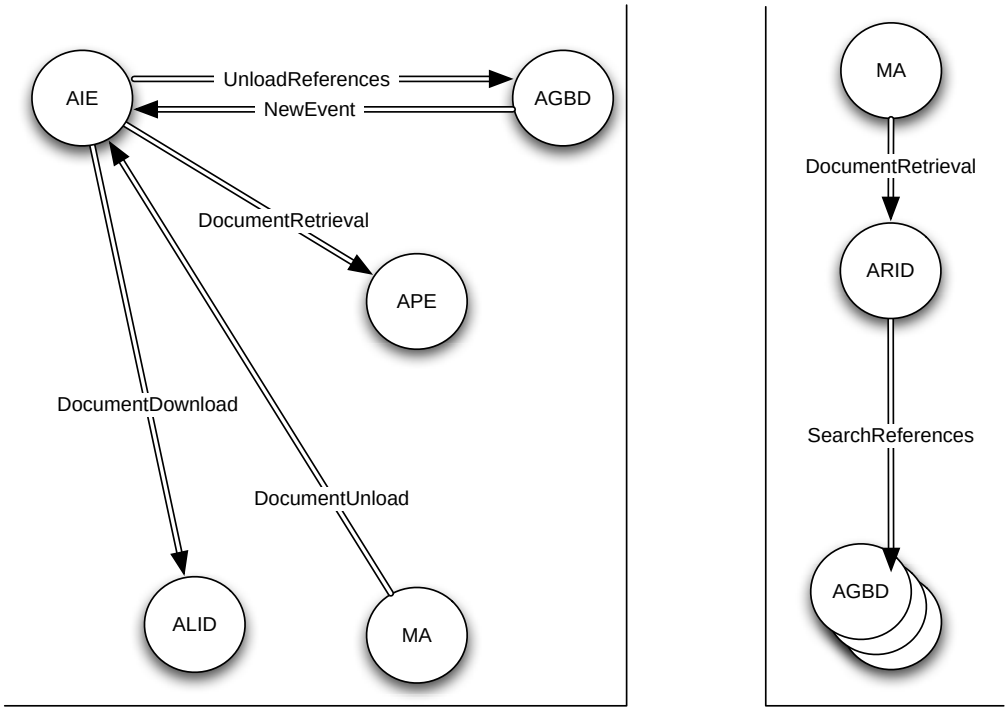


Figura 4.10: Utilització dels serveis dels agents per d'altres agents

Ho pot fer ja sigui perquè ha enviat un missatge sense seguir el protocol o amb una ontologia que no té el receptor. Això farà resetejar tot el protocol.

També en qualsevol moment d'un protocol, aquell que l'ha iniciat pot cancel·lar-lo. Això és possible per que pot ser que ja no estigui interessat en allò que ha iniciat. El receptor contestarà a la cancel·lació amb un missatge d'*inform-done*, en el cas de que la interacció hagi acabat satisfactòriament abans de rebre la cancel·lació, o amb un missatge de *failure*, en cas de que no s'hagi completat la tasca per a la qual el protocol estava destinat.

fipa-propose

El protocol d'interacció *fipa-propose* [7] permet a un agent iniciar una proposta cap a uns altres agents receptors d'aquesta, que estaran preparats per rebre propostes. Els receptors de les propostes podran acceptar o no la proposta.

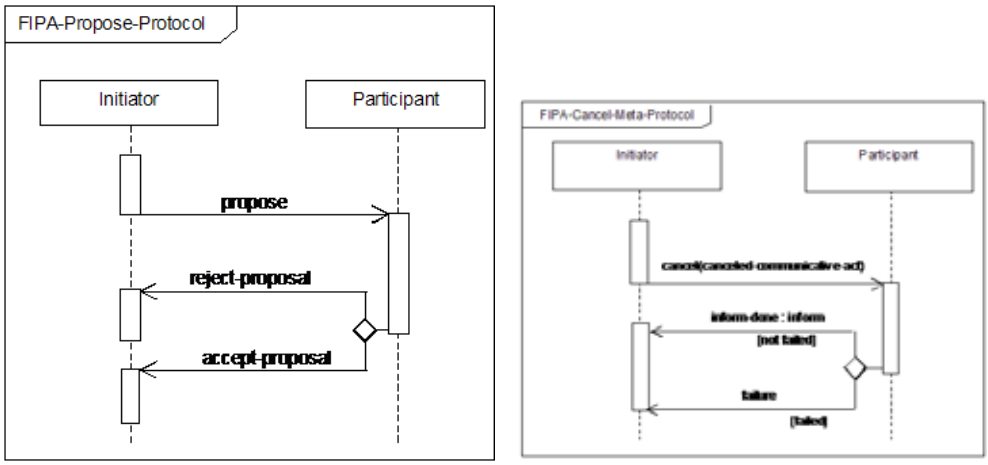


Figura 4.11: FIPA Propose Protocol

fipa-request

El protocol d'interacció *fipa-request* [8] permet a un agent fer una petició a un altre per a que faci una acció. El receptor decideix si accepta (missatge d'*agree*) o no (missatge de *refuse*) aquesta petició.
Si les condicions fan que es necessiti informar a l'emissor, s'enviarà després del missatge d'*agree* un missatge d'*inform*. Aquest informarà de que ha acabat correctament i, opcionalment, podrà incloure en el mateix missatge el resultat d'haver realitzat aquesta petició. En cas de que falli, s'enviarà un missatge de *failure*.
En el cas de que entre el missatge d'*agree* i el missatge d'*inform* hi hagués un temps petit, el missatge d'*agree* es podria ignorar i enviar directament el missatge d'*inform*.

4.8.2 Ontologies

Les ontologies són necessàries per l'enteniment dels agents, definint un context amb el qual s'entenen. Ara que ja hem vist els dos tipus d'esdeveniments que hi poden haver, queda bastant clar que haurem de construir dos ontologies: una primera pels esdeveniments de cerca de documents i una segona pels esdeveniments de descarrega de documents.

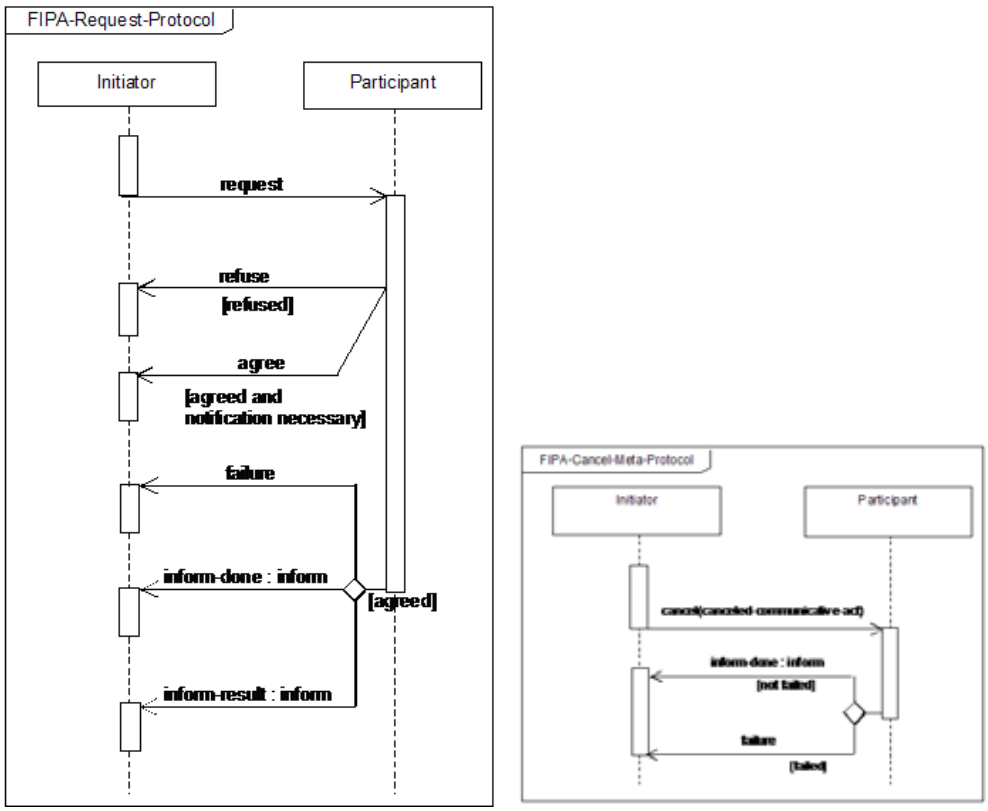


Figura 4.12: FIPA Request Protocol

Document Retrieval Ontology

Aquesta ontologia està destinada a entendre una petició de cerca de documents. Quan vulguem fer una petició d'aquest tipus de forma síncrona haurem de seguir un protocol de *request* explicat al subapartat anterior. Quan fem la petició (missatge *request*) omplirem el missatge utilitzant el primer predicat d'aquesta ontologia. Per les respostes *accept* o *refuse* utilitzarem un missatge buit. I per respondre amb un inform omplirem el missatge utilitzant el segon predicat, que contindrà tota la llista de referències trobades. Utilitzarem aquest mode, per exemple, quan passem la petició des de l'agent mòbil a l'ARID, esperant el primer una resposta (llista de referències) per part d'aquest últim.

Quan vulguem fer una petició de cerca de documents de forma asíncrona haurem de seguir el protocol de *propose*. L'única diferencia amb la forma síncrona, serà

que no esperarem cap resultat, és a dir, cap missatge d'*inform*. La petició (missatge *propose*) contindrà el primer predicat com a ontologia i els missatges d'*accept* o *reject* seran buits. Utilitzarem aquest mode, per exemple, quan passem la petició des de l'AIE a l'APE. Ho farem d'aquest mode perquè l'AIE només esperarà que l'APE hagi rebut la petició no que li contesti amb la llista de referències. Això últim ho faran els agents mòbils quan tornin de visitar totes les plataformes, per això diem que es fa de forma asíncrona.
Ara bé, com descarregaran els agents mòbils totes les referències cap a l'AIE? I aquest últim com farà el mateix cap a l'AGBD? Doncs bé, utilitzarem el protocol *propose* i al mateix missatge de *propose* enviarem tota la llista de referències que tinguem, és a dir, omplirem el missatge amb el segon predicat. El missatge de resposta d'*accept* o *reject* serà buit.

Document Retrieval Predicate

DOCUMENT_RETRIEVAL_PREDICATE (PREDICATE)
DOCUMENT_RETRIEVAL_DESCRIPTION (CONCEPT)
PATIENT_ID (STRING)
OWNER (STRING)
FROM_DATE (DATE) [OPCIONAL]
UNTIL_DATE (DATE) [OPCIONAL]
TYPE (STRING) [OPCIONAL]

Aquest predicat ens servirà per fer peticions de cerca de referències. Té un *concept* que és la petició, el qual conté tots els camps necessaris per fer la cerca. Un camp "PATIENT_ID" que contindrà l'identificador global únic del pacient. Un camp "OWNER" que ens identificarà el personal mèdic que fa la petició important com veurem a l'apartat d'aspectes legals. Uns camps "FROM_DATE" i "UNTIL_DATE" que ens restringiran les cerques a dates compreses dins d'aquest rang. I finalment, un camp "TYPE" que ens restringirà la cerca a referències que siguin només d'aquell tipus. Aquests últims tres camps són opcionals i ens serviran per afinar cerques de referències.

La petició que portin els agents mòbils durant el seu recorregut seran aquest mateix predicat, podent-lo utilitzar directament a l'hora de fer el *request*.

Document Info Predicate

```
DOCUMENT_INFO_PREDICATE (PREDICATE)
    DOCUMENT_INFO_LIST (CONCEPT) [Array]
        DOCUMENT_INFO_DESCRIPTION (CONCEPT)
            ID (STRING)
            ID_DOCTOR (STRING)
            NUMSAN (STRING)
            URI (STRING)
            DATE (DATE)
            TYPE (STRING)
            DESCIPTION (STRING)
            PLATFORM (STRING)
```

Aquest predicat ens servirà per retornar referències de les peticions de cerca. El predicat conté una llista de *concepts* on cada membre tindrà un *concept* que serà la descripció d'una referència. D'aquesta manera podrem enviar més d'una referència a la vegada en el mateix missatge. Les dades d'una referència seran: un camp "ID" que serà un identificador global únic de referència. Aquest identificador global únic el construirem a partir de l'identificador únic global de la plataforma (centre mèdic) on estigui el report, més l'identificador únic intern que aquella plataforma utilitzi com a clau primària per a aquella referència o report. D'aquesta manera aconseguirem l'identificador global únic que busquem; també tindrem el camp "ID_DOCTOR" que ens identificarà quin professional de la medicina va encarregar o va fer el report, és a dir, el responsable amb el qual hauríem de contactar si es tingués cap dubte; un camp "NUMSAN" que identificarà al pacient de manera única global (coincidirà amb el camp "PATIENT_ID" de la petició); un camp "URI" que contindrà la direcció d'on ens podrem descarregar el report; un camp "DATE" que contindrà la data en que es va fer el report; un camp "TYPE" que contindrà el tipus de report (radiografia, anàlisis de sang, ecografia,

etc); un camp “DESCRIPTION” que podrà contenir una petita descripció sobre el report: perquè es va demanar fer-ho, comentaris del personal mèdic que el va analitzar, tractament posat, resposta al tractament posat, entre d’altres aspectes que es creguin rellevants; i finalment, un camp “PLATFORM” que contindrà l’identificador de la plataforma on pertany el report.
Totes les referències noves que vagin recollint els agents mòbils durant el seu recorregut per les plataformes s’aniran afegint a aquest predicat que tindran els agents mòbils emmagatzemat a dins seu. Quan acabin el recorregut li enviaran directament aquest predicat, amb totes les referències recollides, a l’AIE de la seva plataforma origen utilitzant el protocol *propose*. L’AIE recollirà totes les referències de tots els agents mòbils referents a un mateix esdeveniment. Quan les tingui totes les enviarà en un únic predicat (d’aquest tipus) i missatge a l’AGBD, utilitzant també el protocol *propose*, per tal de que aquest últim les guardi a la base de dades.

Document Download Ontology

Aquesta ontologia l’utilitzarem per fer peticions per descarregar referències. Per fer les peticions utilitzarem el protocol *propose* omplint el primer missatge (missatge *propose*) amb el *action*. El missatge de resposta d’*accept* o *reject* serà buit com a l’ontologia anterior.

Document Download Action

```
DOCUMENT_DOWNLOAD_ACTION (ACTION)
    DOCUMENT_DOWNLOAD_DESCRIPTION (CONCEPT)
        DOCUMENT_ID (STRING)
        URI (STRING)
        OWNER (STRING)
```

Aquest predicat representarà una petició de descarrega de referència i tindrà els següents camps: un camp “EVENT_ID” que ens servirà per identificar dins de la nostra plataforma sota quin esdeveniment clínic s’ha demanat la petició de

descarrega del document / referència; un camp “DOCUMENT_ID” que contindrà la direcció d’on hem de descarregar el document (URI); i un camp “OWNER” que ens servirà per identificar qui ha demanat la petició de descarrega d’aquella referència, necessari per complir els aspectes legals del projecte.

4.9 Aspectes Legals

La Llei Orgànica de Protecció de Dades de Caràcter Personal (LOPD [14]) ens obliga a identificar i registrar qui accedeix a una dada mèdica i també en el cas de que faci una modificació. Com ja hem dit anteriorment, seguirem aquesta llei ja que s’emmarca dins la directiva europea i farà que el projecte sigui compatible en el marc de la Unió Europea. Per tal de complir la llei, tots els missatges de petició de cerca de l’historial d’un pacient, així com els de descarrega inclouran un camp que hem vist a l’apartat d’ontologies anomenat *owner*. Aquest camp servirà per identificar qui fa la petició i d’aquesta manera poder registrar-ho a la base de dades d’on llegim la dada. Seguir la llei també implicarà la protecció de la interfície a través d’un nom d’usuari amb identificació de qualsevol tipus: biomètric, amb paraula de pas, etc i el que és més important: els agent actuaran en nom d’una persona concreta. L’agent, al portar l’identificador de qui ha fet la petició, actuarà en nom seu i totes les accions que faci seran responsabilitat de qui ha demanat la petició. Les dades han d’estar protegides per a evitar atacs de falsificació o impersonació.

4.10 Agents Mòbils

Com s’ha pogut observar a la figura de l’esquema de comunicació, el sistema multi-agent comptarà amb una *pool* d’agents. Aquest “dipòsit” d’agents mòbils es podrà configurar per tenir una determinada quantitat d’agents mòbils disponibles en cada moment segons la necessitat de l’aplicació. Si el dipòsit es queda sense agents mòbils, aquests es crearan sota demanda quan es necessitin. L’encarregat de mantenir el dipòsit amb agents i de crear-los si se’n necessiten (sota demanda)

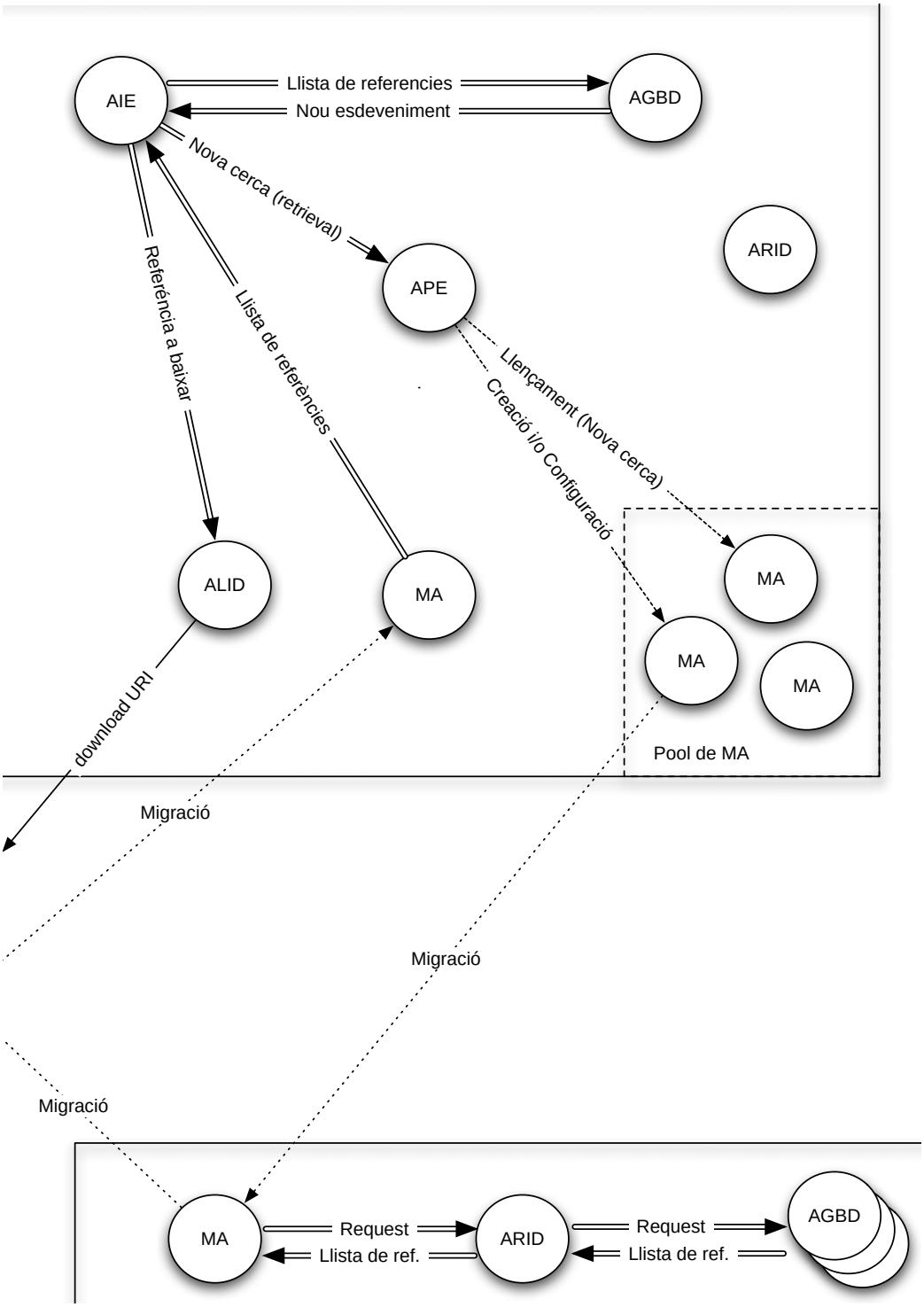


Figura 4.13: Diagrama de missatges ACL

serà l'APE.

Com podem veure a la figura 4.14, la creació d'agents mòbils es fa a una capa inferior del sistema multi-agent. Això és degut a que com classe de Java que és, la instanciació de la classe corre a càrrec de JADE i Java.

La configuració dels agents mòbils també serà a la capa JADE / Java degut a que l'objectiu és posar els paràmetres correctes als agents mòbils. Per tant, aquesta configuració la farem amb crides a mètodes dels agents mòbils enlloc de missatges ACL com a la capa de sistema multi-agent. Els paràmetres a configurar seran la llista de plataformes que hauran de recórrer els agents buscant informació i les dades de la petició de cerca de documents. Quan aquests dos paràmetres estiguin configurats els agents mòbils estaran llestos per poder ser llençats. Tindrem l'opció de que els agents mòbils comencin a migrar quan rebin els paràmetres de configuració, o bé, l'APE els podrà llençar manualment si ho creu convenient.

4.11 Migració

L'*add-on* de migració JIPMS [11] serà necessari pel nostre projecte per tal de que els nostres agents mòbils puguin viatjar d'una plataforma a una altre. Des del punt de vista del disseny influirà a l'hora de llençar la plataforma ja que haurem d'incloure el servei de migració a la comanda d'execució. A més, tindrem una restricció a l'hora de programar i d'escollir les classes que voldrem utilitzar. Les classes que es vulguin utilitzar als agents mòbils hauran de ser serialitzables per tal de que l'agent pugui migrar. També tindrem una altra restricció a l'hora de dissenyar el comportament de l'agent mòbil, ja que haurem d'utilitzar un *behavoiur* de tipus màquina d'estat.

4.12 Múltiples peticions

Si seguim l'esquema actual veurem que no està preparat per atendre més d'una petició alhora. Per posar un exemple: mentre l'agent ALID estigui baixant un document remot no podrà atendre al mateix temps una altra petició i començar a

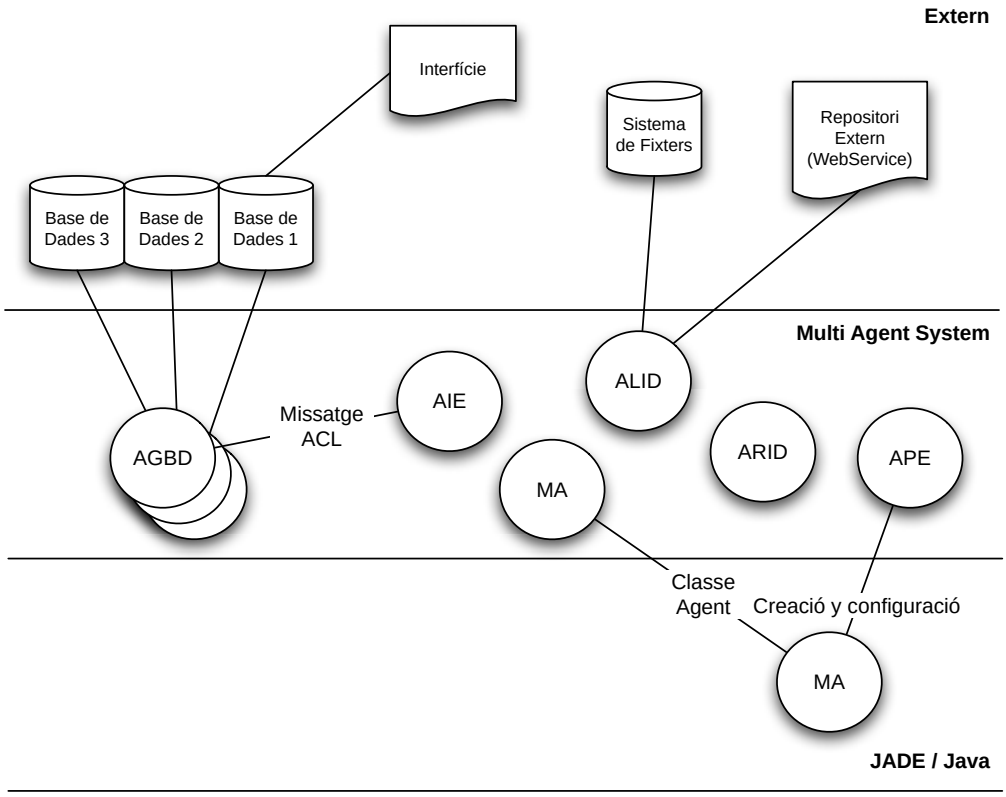


Figura 4.14: Diagrama de capes

baixar-la. És a dir, no podrem tenir més d'una petició a gairebé cap agent segons els sistema plantejat, tots estaran ocupats i no podran atendre a més peticions. Aquest és un esquema seqüencial, les peticions s'aniran posant en cua i s'aniran atenent en quant els agents vagin acabant les anteriors.

Per tal de millorar aquest punt vam decidir utilitzar fils d'execució (*threads*). Els *behaviours* dels agents habitualment s'executen de manera seqüencial com hem pogut veure al capítol 2. Però existeix una altra manera d'executar *behaviours* i es tracta de llençar-los com *threaded-behaviours*. Aquests *behaviours* enlloc de ficar-se a dins la cua de *behaviours* i executar-se seqüencialment, s'executaran tots alhora permetent, per exemple, poder atendre múltiples peticions al mateix temps. A més, podrem afegir tants com vulguem en el moment que vulguem, permetent-nos d'aquesta manera adaptar-nos a l'escalabilitat donada. Per

exemple, quan rebem una petició a un *behaviour*, abans d'atendre-la llençarem un nou *threaded-behaviour* per tal d'atendre una altra possible petició que arribi mentre s'està atenent aquesta i així amb tots els *behaviours*. Quan aquest *behaviour* acabi d'atendre la petició s'eliminarà, per tant, els recursos consumits sempre seran els mínims possibles.

4.13 Descarrega de documents

En aquest apartat dissenyarem com seran els repositoris on tindrem els documents per tal de que les altres plataformes puguin accedir a ells. L'objectiu és dissenyar un sistema robust i que ens ofereixi seguretat i interoperabilitat a través de la xarxa. Per fer això utilitzarem un sistema molt utilitzat avui dia: els Web Services. Els Web Service es comuniquen entre client i servidor utilitzant missatges XML els quals es podran xifrar i signar utilitzant serveis que ens proporciona el servidor web service així com també es pot utilitzar canals segurs com https. A més, permeten interaccions per diferents tipus d documents, integració en l'entorn actual, parametrització de les consultes (descarregues parcials), entre d'altres característiques. Per tant, s'adapta a les nostres necesitats i en farem ús d'ell. Més exactament, escollirem la solució de l'Apache Software Foundation [17]. Com a opció de disseny alternativa es podria escollir un servidor web.

4.14 Seguretat

El disseny de la seguretat és un punt molt important al nostre projecte, ja que com hem vist durant aquest capítol, els aspectes legals així ho requereixen. A més, la importància de la integritat d'una dada especifica d'un pacient pot fer salvar la seva vida. Aspectes importants com les infermetats d'un pacient (SIDA, Hepatitis...) són dades que hem de mantenir en secret. Per això també hem de mantenir un accés restringit a aquelles dades a només el personal degudament autoritzat. Per tot això, privacitat, autenticitat i integritat, és necessari implantar un bon sistema de seguretat.

L'esquema de seguretat que s'utilitzarà ja està dissenyat i el disposem a l'article [2]. Aquest esquema fa que els agents mòbils (els únics elements vulnerables de l'esquema, ja que són els únics elements que surten de la plataforma). Assumim que la plataforma no actuarà amb malícia sobre els agents, podrem confiar sempre en ella, pel seu propi interès.

Per tal d'implementar aquest esquema de seguretat ens caldrà un servei criptogràfic per a JADE, en el qual s'està treballant en un altre projecte paral·lel a aquest [22]. L'esquema consisteix en el següent: el sistema criptogràfic generarà una clau privada (Sj) i pública (kj) a cada plataforma. Cada agent mòbil utilitzarà el servei criptogràfic per generar-se una clau privada (Sa) i pública (Pa). L'agent mòbil destruirà la clau privada (Sa) quan marxi de la plataforma origen per tal de que aquesta no sigui compromesa i guardarà la clau pública (Pa) al codi de control de l'agent. La clau pública (Sa) ens servirà posteriorment per a que les plataformes puguin comprovar la signatura de l'agent i d'aquesta manera comprovar que el codi de l'agent és correcte.

El codi de l'agent que no sigui codi de control (d) i una signatura d'aquest codi (s) es xifrarà utilitzant una clau simètrica (r) amb una funció (E). La signatura està generada a partir del hash del codi de l'agent (h(d)) signat utilitzant la clau privada (Sa) de l'agent. Per tant, tot aquest xifratge conjunt (D) es generarà abans de que l'agent deixi la plataforma d'origen i serà immutable durant tot el recorregut de l'agent.

El resultat del xifratge conjunt (D) juntament amb el codi de control de l'agent (C), que no es podrà xifrar ja que és el codi que varia durant el recorregut de l'agent, formaran el nou agent protegit. Però també necessitarem que el codi de control (C) sigui segur, sinó el podrien manipular i fer que l'agent faci accions no desitjades. Per tant, hi haurà una tercera part al nou agent que es composarà del següent: una signatura del codi de control (h(C)) xifrada juntament amb la clau simètrica (r) que hem utilitzat anteriorment per fer una part del nou agent (D). Aquest xifratge es farà amb una funció (E) utilitzant la clau pública de la següent plataforma a la que l'agent mòbil haurà de saltar (kj). D'aquesta manera la pròxima plataforma podrà comprovar que el codi de control és correcte i obtenir la clau

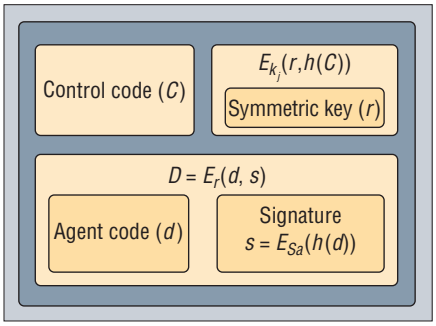


Figura 4.15: Agent auto-protegit

simètrica per desxifrar i comprovar el codi de l'agent (d).
Per tant, utilitzant aquest esquema, que podem veure gràficament a la figura 4.15, protegirem a l'agent de possibles canvis al seu codi per part de tercers. En el cas de que es detectés algun canvi no desitjat (signatura incorrecte) aquest no s'executaria.

En el mateix esquema, proposat a [1], també s'implementa la protecció de les dades mèdiques que aquest agent transporti. Aquest sistema consisteix en xifrar, utilitzant la clau pública de la plataforma d'origen de l'agent mòbil, els resultats obtinguts per l'agent a cadascuna de les plataformes. Amb aquest sistema podrem protegir el contingut dels resultats durant tot el recorregut, però no és del tot suficient perquè es podrien afegir nous resultats falsos o eliminar d'actuals. Per evitar això també haurem de protegir l'itinerari. Cada plataforma afegirà una estructura signada que inclourà:

1. Un hash dels resultats xifrats amb la clau pública de la plataforma origen de l'agent mòbil.

2. Un hash dels resultats de la plataforma anterior xifrats amb la clau pública de la plataforma origen de l'agent mòbil.

3. El nom de la pròxima plataforma a visitar.

Amb aquesta estructura la plataforma d'origen de l'agent mòbil podrà comprovar si ha hagut intervenció de tercers (inserció, eliminació, canvi de itinerari, etc)

durant el recorregut.

A més de tot això, si hem d'intercanviar missatges entre plataformes, utilitzarem el mecanisme "ACL in ACL" i el servei criptogràfic (claus públiques i privades de plataformes i agents) per tal de xifrar el contingut del missatge. D'aquesta manera el contingut no podrà ser accedit per tercers.

Per tant, tindrem un esquema segur que només es podrà trencar si existeix una plataforma maliciosa. Això serà molt poc probable degut a que totes les plataformes actuen en benefici de tota la xarxa mèdica.

Capítol 5

Implementació, proves i discussió

En aquest capítol veurem els detalls i problemes durant la implementació del disseny proposat al capítol anterior. Una vegada acabada la implementació passarem a provar l'aplicació sota diferents escenaris. Provarem la seva capacitat i veurem les seves bondats i defectes. A l'apartat de discussió analitzarem aquestes dades amb més profunditat i comentarem les possibles millores o punts febles.

5.1 Implementació

Com ja s'ha comentat varies vegades, la implementació es va fer utilitzant el llenguatge de programació Java i el *framework* JADE. Seguidament passarem a comentar alguns detalls o problemes, que creiem que són dignes de comentar, que van sorgir durant la implementació.

Com vam comentar a l'apartat de "Múltiples peticions" del capítol de Disseny, vam implementar *behaviours* amb *threads*. Va sorgir un problema però. La versió actual de JADE, quan volem bloquejar un agent per tal de que esperi un missatge, bloquejarà la seva cua de missatges fins que se'n rebi un. Quan utilitzem *threads*, com l'agent estarà dividit en uns quants (un per cada *threaded-behaviour*) i tots s'executaran a la vegada, el bloquejar la cua, causarà problemes i l'agent es quedarà penjat. El problema es podria solucionar bloquejant només el fil d'execució que ho necessiti, enlloc de la cua de missatges. Aquesta solució, però, passa per

modificar el codi de la plataforma JADE.

Un altre detall a l'hora de fer la implementació ha estat els registres. Per tal de que els agents puguin utilitzar una ontologia i llenguatge concret, haurem de registrar-los a l'agent. Aquest registre es pot fer al *setup* de l'agent un sol cop. Però amb els agents mòbils tenim una excepció: haurem de fer el registre cada vegada que canviïn de plataforma.

El codi ha sigut comentat degudament i s'ha seguit la forma JavaDoc per tal de fer-ho. D'aquesta manera serà més fàcil entendre'l i poder fer ampliacions futures.

Per programar el projecte vam utilitzar el SDK *(Software Development Kit)* "Eclipse" [19] i per fer els diagrames i esquemes necessaris vam fer servir l'aplicació "OmniGraffle" [20].

Vam decidir incloure els comportaments (*behaviours*) dins dels agents com a classes privades.

Es va utilitzar el sistema de gestió de base de dades "One Dolar DataBase" [5] per implementar la base de dades de prova. La programació de la base de dades es va fer utilitzant el llenguatge de programació SQL. Es va seguir el disseny comentat al capítol anterior.

5.2 Proves

Durant el transcurs del projecte vam haver de fer diverses proves per tal d'ajudar-nos a decidir entre una decisió o una altra de disseny.

Una de les proves que vam fer ens va ajudar a decidir que era millor, enviar un missatge ACL gran, o bé, molts de petits. Aquesta discussió va aparèixer derivada de que al servei de migració, quan un agent supera els 25 kB la seva velocitat de transferència decau exponencialment. Per tant, vam estar pensat si seria millor, enviar tots els resultats de buscar referències en un sol missatge, o bé, enviar-ne uns quants de petits. Els resultats de les proves van demostrar que el temps era més baix a l'enviar el missatge gran, encara que aquest sigues molt gran (10 MB). Vam demostrar que això es degut a que el problema de la baixada de velocitat

exponencial a partir de 25 KB només es dona a l'intercanvi de missatges inter-plataforma. Com els nostres intercanvis de missatge són intra-plataforama no teníem aquest problema.

Una altra de les proves realitzades va consistir en implementar un servidor web en Java. Això ho vam fer per veure la viabilitat de poder integrar la interfície dins del sistema multi-agent juntament amb la base de dades. Aquest servidor web fet en Java oferia la interfície, per https, de mode segur. A més, intercanviava missatges amb l'AGBD per tal de mostrar-nos a la interfície les dades dels esdeveniments clínics i reports. Aquesta idea es va descartar ràpidament quan vam corregir l'error comés inicialment al situar la interfície dins del sistema multi-agent. Aquesta ha d'estar fora ja que les seves insercions o consultes a la base de dades són independents del sistema multi-agent. El sistema multi-agent no detecta cap canvi quan la interfície fa una inserció o consulta, només quan l'AGBD fa la petició. Per tant, és un recurs exterior al sistema multi-agent. Aquest és el motiu principal de perquè el vam situar fora, a més a més, la interfície fa insercions i consultes directament sobre la base de dades.

La prova més important, però, va ser la prova d'implementació del projecte. Un cop acabada la implementació vam configurar una sèrie de màquines per provar-la. La configuració va ser la de 6 màquines posades en xarxa de forma que cadascuna sigues un centre mèdic amb una plataforma. Cadascuna tenia la seva base de dades pròpia amb els seus registres de pacients, esdeveniments clínics i reports. Es van posar en marxa totes i es van configurar per a que sapiguessin de l'existència de les altres cinc. Es van llençar esdeveniments dels dos tipus des de totes les plataformes obtenint resultats satisfactoris en tots els casos. Les proves van anar augmentant de complexitat per tal de veure el límit del programa. Es van realitzar múltiples esdeveniments simultanis des de totes les plataformes i, de nou, tots els resultats van ser satisfactoris.
Una vegada provat això, vam provar la tolerància a fallades del sistema. El programa respon correctament enfront a plataformes que no responen. L'agent salta a la següent plataforma i acaba el camí correctament. Sota altres circumstàncies com enviar missatges amb continguts erronis el sistema també va respondre cor-

rectament.
On si que va ser vulnerable va ser a la prova de pèrdua de dades per la xarxa, és a dir, missatges que no arriben al destinatari. La falta d'un mecanisme de *timeout* adequat va fer que els agents es quedessin esperant un missatge que no arribaria. Necessitaríem una millora d'aquest mecanisme que comentarem al següent apartat.
Val a dir que totes les bases de dades de totes les plataformes disposaven d'exemple de tots tipus per tal de que la prova fos el més real possible.

5.3 Discussió

Ara que ja hem vist com hem implementat el projecte, els problemes que han sorgit i les proves realitzades durant i després de la implementació, passarem a fer una discussió del resultat.

Tot i que el sistema és suficientment robust, no ho és tant com ho voldríem. No és suficientment robust a la pèrdua de missatges. El mecanisme actual implementat fa que si el missatge que un agent envia no és rebut per el destinatari, l'emissor d'aquest missatge rebrà un avís d'aquest fet. Això fa que l'agent emissor pugui tornar a enviar-ho i fer uns quants intents abans de deixar d'intentar-ho i marcar-ho com a error.
Però el mecanisme que ens farà fallar el sistema multi-agent serà si estem esperant un missatge que no arribarà mai. Això pot ser perquè l'agent emissor d'aquest missatge no l'envii o perquè es perdi contínuament (problemes de xarxa). Aquest fet es pot arreglar implementant un sistema de *timeouts* millor que faci tornar a començar el protocol, que en aquell moment esta utilitzant l'agent, si durant un temps prudencial no es rep cap resposta de l'altre agent.
Val a dir que la manera en que falla el sistema no és quedant-se blocat sinó en que consumirem recursos innecessaris quan un error d'aquest tipus es produexi. Gracies al sistema de *threads* implementat el sistema podrà seguir treballant. El problema serà que tindrem un *thread* esperant indefinidament un missatge que no arribarà, però sense consumir CPU, ja que estarà en estat *sleep*.

Un mecanisme que també caldrà implantar serà un sistema de reintents per aquelles plataformes a les quals no s'hagi pogut arribar, o bé, hi hagi hagut un error amb l'ARID. Per tant, ens caldria un sistema a l'agent, una mena de llista, que emmagatzemes totes aquelles plataformes on hi hagi hagut un error. Al tornar a la plataforma inicial, aquesta llista es passaria a l'APE per tal de que pugués tornar a enviar altres agents més tard per tornar-ho a intentar.

A part d'aquest dos errors el sistema s'ha comportat correctament sense cap falla a proves de 6 plataformes amb peticions continues de tots dos tipus d'esdeveniments. Tot de manera automatitzada.

Com ja hem parlat moltes vegades al llarg de la memòria, la seguretat és un punt en el que volem insistir. La falta d'un servei criptogràfic actualment a JADE ens va fer que no poguéssim implementar l'esquema de seguretat en aquesta prova de concepte. Malgrat això, el disseny de l'esquema de seguretat està fet i només s'hauria de seguir per tal de tenir la prova de concepte funcionant amb seguretat quan el servei criptogràfic estigues disponible.

El *webservice* tampoc es va acabar implementant. La nostra prioritat era acabar la prova del concepte amb la màxima robustesa i les característiques més importants possibles d'implementar. La petició cap a un servidor web, actualment implementada en la prova de concepte, es podria substituir per una implementació del *webservice* sense afectar a la resta.

Capítol 6

Conclusions

Ara que ja hem fet tot el desenvolupament complet de l'aplicació (estudi, anàlisi, disseny, implementació i proves), i tot just arribem a l'últim capítol, és hora de fer un resum.

A l'inici d'aquest projecte ens hem proposat l'objectiu de realitzar el desenvolupament d'una aplicació que satisfaci unes necessitats del món mèdic actual: la compartició d'informació i la cooperació més eficaç possible entre centres.

Primerament hem analitzat l'estudi previ fet a l'article *Secure Integration of Distributed Medical Data Using Mobile Agents* [1]. Hem vist quin és l'estat de l'art d'aquest estudi així com l'actualitat al món de les aplicacions mèdiques, sobretot les basades amb agents. Ens hem endinsat en el món dels agents per poder entendre'l millor i poder preparar-nos per donar els següents passos. Hem estudiat com funcionen els agents, els seus estàndards [10], la plataforma que hem d'utilitzar [13], la migració dels agents a d'altres plataformes [11] i altres aspectes dels agents.

Un cop fet tot això hem fet un anàlisi de les necessitats. En aquest anàlisi hem vist tots els requisits del projecte, tant els funcionals, com els no funcionals: interfícies, requisits de memòria, requisits de plataforma *hardware*, requisits legals, etc; i hem fet un recull d'ells per poder fer el següent pas: el disseny.

Hem fet un disseny que satisfà tots els requisits esmentats a l'anàlisi. Hem tingut en compte totes les alternatives possibles a l'hora de fer cada part del disseny

i hem anat posant els avantatges i inconvenients en cadascuna d'elles. Això ens he permès escollir la millor opció possible en cada cas.

Una vegada acabat el disseny amb les opcions més avantatjoses pel projecte, hem fet una prova de concepte d'aquest. Hem escollit els conceptes de disseny que més ens interessaven implementar, d'entre tot el disseny, per tal de tenir una prova de concepte totalment funcional. Això ha estat així a causa de que la implementació de tot el disseny portaria molt de temps, ja que és un projecte englobat en un altre projecte molt més gran a llarg termini. Hem vist els detalls de la implementació i com hem hagut de deixar fora els aspectes de seguretat i *webservice* per falta de recursos. Finalment hem aconseguit acabar-la. El resultat de la implementació ha estat un sistema totalment funcional amb aquelles característiques del disseny que hem cregut prioritàries per tal de portar-lo a la pràctica.

Una vegada amb la implementació acabada, l'hem verificat a fons i hem vist les seves bondats i limitacions. Hem vist que les proves realitzades, de manera exhaustiva, han donat resultats molt satisfactoris. També hem vist les limitacions en quant a tolerància a fallades (bloquejos d'agents en alguns casos especials, per exemple) i detalls no implementats com el de la seguretat o el temps màxim d'espera a l'intercanvi de missatges.

Una vegada amb totes aquestes dades hem intentat ser crítics per tal d'extreure'n conclusions.

Tots aquests passos s'han seguit segons el calendari establert al diagrama de Gantt que hem planejat al principi del projecte.

Per tant, tots els objectius que ens hem proposat a l'inici d'aquest projecte els hem assolit sense complicacions.

6.1 Millores de l'esquema i implantació a hospitals

La implementació feta s'ha fet amb l'objectiu de tenir una prova de concepte. Per tant, hi ha conceptes dissenyats que no hi són a la implementació. Això és degut a que la implementació d'aquells conceptes és massa complexe o llarga com per tenir-la en compte en el temps i recursos dedicats al projecte. Tot i això vam

creure necessari dissenyar-los per una possible implementació completa futura, ja que aquest projecte s'emmarca dins d'un altre projecte molt més gran a llarg termini.

A més de tot això, també existeixen alguns aspectes externs al projecte (no dissenyats ni implementats per nosaltres), però necessaris per a la seva execució en un escenari real, que tenen algunes limitacions que cal superar.

Tot això ens dona peu a obrir línies d'investigació futures que ens permetran millorar l'aplicació i afegir-hi noves característiques. Aquestes línies són:

Seguretat Un camp que ha quedat obert ha sigut el de la seguretat. JADE no ofereix actualment cap mecanisme per poder implementar l'esquema de seguretat presentat a [1], basat en [2], que sigui compatible amb el servei de migració. Paral·lament a la realització d'aquest projecte, un altre projecte de fi de carrera (*Desenvolupament d'un servei criptogràfic per a la plataforma JADE*) [22] ha estat realitzant un servei de JADE que permetria fer això. Com que els projectes s'han desenvolupat alhora, no s'ha pogut fer una integració. Com a possible ampliació seria viable, i d'aquesta manera millorar la seguretat de l'actual implementació.

Tolerància a fallades Tot i que s'ha fet un gran esforç per disminuir al màxim la possibilitat de fallades i d'errors durant l'execució, encara existeixen situacions, sota determinades circumstancies, que podrien fer que l'aplicació no funcionés correctament. És una aplicació de caire complexe i que conté molt agents. Cada agent fa més d'una tasca i tots ells són autònoms, és a dir, és difícil detectar errors i corregir-los. En part, això és degut a que no existeixen eines adequades de depuració per sistemes multi-agents. Una ampliació possible seria implementar un sistema més robust de tolerància a fallades, ja que l'ambient en que aquestes plataformes s'han d'executar així ho requereix.

Fragmentació Un dels problemes de la migració consisteix en que quan la instància d'un agent es fa gran, més de 25 kB, el temps que triga l'agent en migrar d'una plataforma a una altra creix exponencialment. Això és un pro-

blema, degut a que quan tinguem un agent mòbil que vagi carregat amb moltes referències es mourà molt lent o inclús podria causar *timeouts*. Aquesta és una altra línia d'ampliació futura, ja no envers el projecte sinó en la millora del servei de migració. Aquesta millora de JADE i més en concret del servei de migració ens afectaria directament en el rendiment de la nostra aplicació.

A més d'aquests conceptes també haurem de tenir en compte tots aquells explicats a l'apartat de discussió del capítol anterior. Per fer la implantació a centres mèdics reals, fora de l'àmbit de prova, necessitarem una implementació més avançada de la que es disposa actualment. Seguint tot el disseny realitzat a aquest projecte per tal de continuar amb la implementació i, a més, aplicant les millores i/o ampliacions esmentades, s'aconseguirà una aplicació per poder implantar a qualsevol centre mèdic en una xarxa global.

Amb aquest projecte hem posat el nostre petit gra de sorra per ajudar a millorar el sistema mèdic actual, un dels nostres principals objectius. Gràcies a aquest projecte he aprés molt. El haver de fer tots els passos, des de l'estudi fins a la implementació, m'ha permès veure tots els detalls i dificultats de cadascuna de les fases de desenvolupament. A més, la preparació autònoma necessària per afrontar el projecte ha sigut molt enriquidora. El projecte és gran perquè abasta molt camps, no només els dels agents sinó també el de la seguretat, bases de dades, anàlisi, interfícies, etc. Gràcies a això he pogut aplicar molts conceptes adquirits durant la carrera a diferents assignatures. En definitiva, aquest projecte ha sigut un molt bon punt i final de la carrera.

Bibliografia

[1] Vieira-Marques, P.M.; Robles, S.; Cucurull, J.; Cruz-Correia, R.J.; Navarro, G.; Marti, R.; “Secure Integration of Distributed Medical Data Using Mobile Agents”, *Intelligent Systems, IEEE*, Volume 21, Issue 6, Nov.-Dec. 2006 Page(s): 47 - 54

[2] Ametller, J.; Robles, S.; Ortega-Ruiz, J.A.; “Self-protected mobile agents”, *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.* 2004 Page(s):362 - 367

[3] Basso, A.; *MAID Mobile. Recolha de Informação Clínica baseada em agentes móveis.* Departamento de Ciència de Computadores. Facultade de Ciències de Universidade de Porto, 2006

[4] IEEE; *Recommended practice for software requirements specifications.* 20 Oct. 1998

[5] Daffodil Software Ltd.; *One $ Database.* <http://www.daffodildb.com/one-dollar-db.html>

[6] Unió Europea; *Targeta Sanitària Europea* <http://ec.europa.eu/employment_social/healthcard/index_en.htm>

[7] Foundation for Intelligent Physical Agents (FIPA); *FIPA Propose Interaction Protocol Specification.* SC00036H. 2002/12/03 <http://www.fipa.org/specs/fipa00036/index.html>

[8] Foundation for Intelligent Physical Agents (FIPA); *FIPA Request Interaction Protocol Specification.* SC00026H. 2002/12/03 <http://www.fipa.org/specs/fipa00026/index.html>

[9] English Wikipedia; *Software agents* <http://en.wikipedia.org/wiki/Software_agent>

[10] Foundation for Intelligent Physical Agents (FIPA); *Specifications* <http://www.fipa.org/repository/standardspecs.html>

[11] Cucurull, J.; *JADE Inter-Platform Mobility Service* <http://sourceforge.net/projects/jipms>

[12] IEEE; *Intelligent Systems. Putting AI Into Practice* November/December 2006, vol.21

[13] TELECOM ITALIA; *Java Agent DEvelopment Framework (JADE)* <http://jade.tilab.com/>

[14] Gobern Espanyol; Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal <https://www.agpd.es/upload/Canal_Documentacion/legislacion/Estatal/Ley%2015_99.pdf>

[15] Sourceforge.net; *Gantt project* <http://ganttproject.biz/>

[16] Sun Microsystems, Inc.; *Java 2 Platform Standard Edition 5.0* <http://java.sun.com/j2se/1.5.0/docs/api/>

[17] Apache Software Foundation; *Apache Axis2*. Web service framework. <http://ws.apache.org/axis2/>

[18] Eckel, B.; *Thinking in Java.* Third Edition. Prentice-Hall, December 2002

[19] The Eclipse Foundation; *Eclipse SDK* <http://www.eclipse.org/>

[20] The Omni Group; *OmniGraffle* <http://www.omnigroup.com/applications/omnigraffle/>

[21] IETF; *Lightweight Directory Access Protocol*. RFC 1777 <http://tools.ietf.org/html/rfc1777>

[22] Bigas J.; *Desenvolupament d'un servei criptogràfic per a la plataforma JADE*. UAB, Juny 2007.

Firmat: Abraham Martín Campillo

Bellaterra, Juny de 2007

Resum

En aquest projecte es desenvolupa en totes les seves fases (estudi, anàlisi, disseny, implementació i proves) l'aplicació MedIGS. MedIGS és una aplicació destinada a satisfer unes de les necesitats actuals del sistema sanitari, la compartició d'informació i la màxima coordinació possible, utilitzant una tecnologia novedosa: els agents i més concretament, els agents mòbils. Gràcies a aquesta tecnologia aconseguirem una integració segura de dades mèdiques distribuïdes. Està previst fer una prova pilot a Portugal, a partir dels resultats d'aquest projecte.

Resumen

En este proyecto se desarrolla en todas sus fases (estudio, análisis, diseño, implementación y pruebas) la aplicación MedIGS. MedIGS es una aplicación destinada a satisfacer algunas de las necesidades actuales del sistema sanitario, la compartición de información y la máxima coordinación posible, utilizando una tecnología novedosa: los agentes y más concretamente, los agentes móviles. Gracias a esta tecnología conseguiremos una integración segura de datos médicos distribuidos. Está previsto hacer una prueba piloto en Portugal, a partir de los resultados de este proyecto.

Abstract

In this project we have developed the MedIGS application, along all its stages (research, analysis, designing, implementation, and testing). MedIGS is oriented to meet some important requirements of current healthcare systems, such as distributed information gathering and cooperation among medical centers. To achieve this, we have used agent technology, and more specifically, mobile agents. By using this technology it is possible now to securely integrate distributed medical data. It is planned to run a pilot test on some Portuguese hospitals based upon the results of this project.